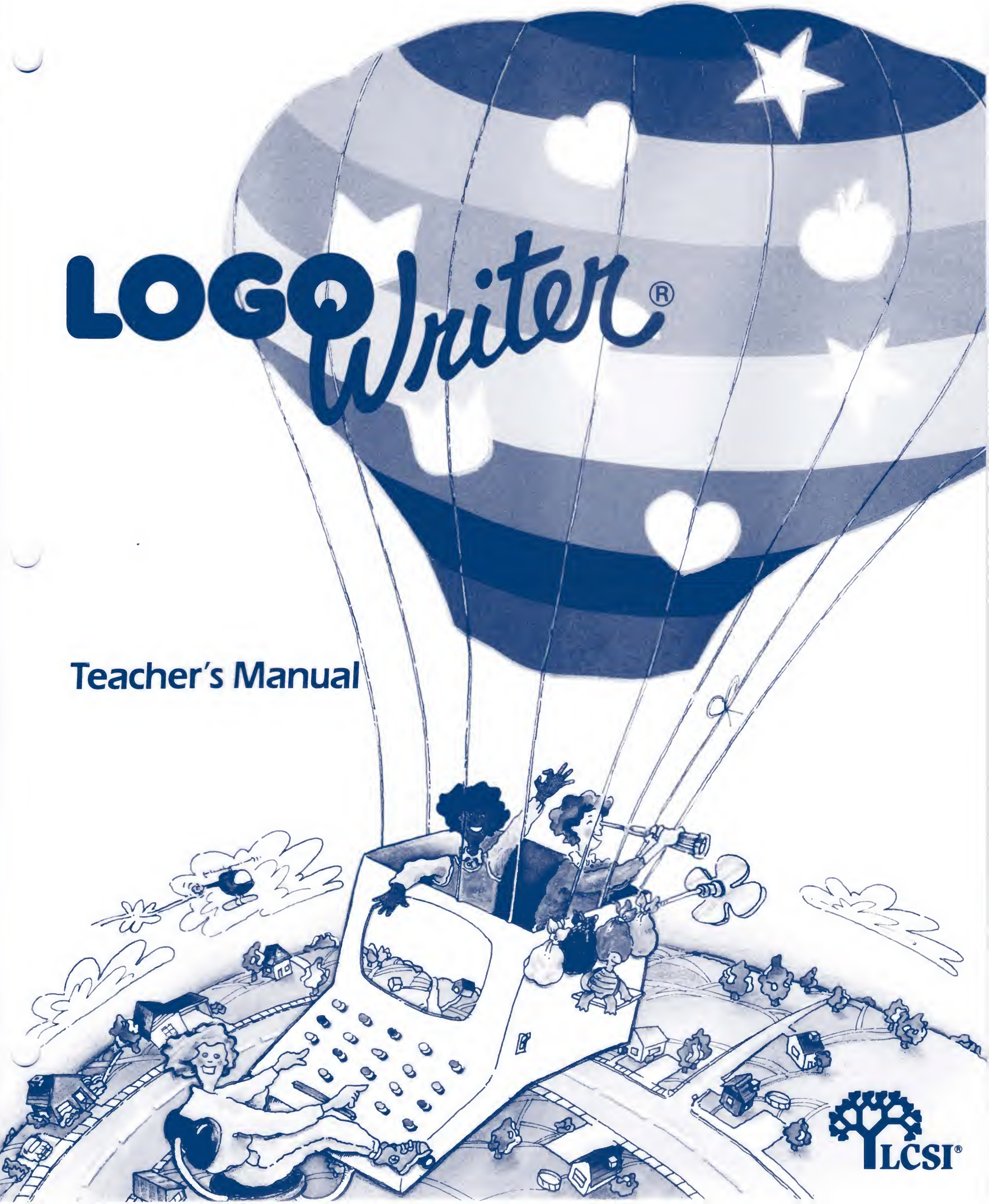
LOGO Writer®

Teacher's Manual



LCSI®

LogoWriter

Teacher's Manual

by Logo Computer Systems Inc.

LogoWriter is an educational product made available to schools from Logo Computer Systems Inc.

Legal deposit, 2nd semester 1986
3rd revision, June 1989

ISBN 2-89371-004-2
ISBN 2-89371-006-9

LogoWriter Team

This page only partially reflects the diversity of tasks that each LogoWriter team member performed during the project.

Software Concept
Seymour Papert
Brian Silverman

Authors:
Teacher's Manual
Sharnee Chait
Judy Le Gallais

Project Booklets
Susan Fischer

Activity Cards
Eric Brown
Alain Tougas

Reference Guide
Eric Brown
Susan Fischer

Contributing Editors
Sharnee Chait
Judy Le Gallais
Nicole Michaud

Design Concept and Illustrations
Le Groupe **Flexidée**

Graphic Design and Layout
Jean-François Dorval
Gabriel Landry
Richard Lavigne
Julien Perron

Production
Lise van Chestein
Brigitte Fréchette

Software Engineers
Apple IIe, IIc Version
Mario Bergeron
Mario Carrière
Mamadou Billo Diallo

LogoWriter IBM PC, PCjr Version
Pascal Bekaert
Annick Hernandez

Programmers/Testers
Paul Legault
René Yelle

Educational Research
Nicole Michaud
Kiyoko Okumura
Michael Tempel

Project Management
Effie Maniatis
Barbara Mingie
Renaud Nadeau

Table of Contents

Introducing LogoWriter 7

Chapter 1
A Walk Through Your LogoWriter Kit 9

Chapter 2
The Project Approach 17

Chapter 3
Organizing the Classroom 23

Chapter 4
On Your Mark, Get Set, Go 27
Making a Birthday Card 29
Writing a Fan Letter 30
Street Scene 30

Chapter 5
Word Adventures 35
Crossword Puzzle 37
Adventure Story 41
Secret Code 43

Chapter 6
Words Come Alive! 49
Pun Fun 51
Row, Row, Row Your Boat – A Music Video 55

Chapter 7
You and Your Friends 59
Walking Through Your Neighborhood 61
Dice 4 65
Leaping Turtle 70

Chapter 8
News, Views, and Information 75
Lights, Camera, Action 77
Traveling Through Time 79

Chapter 9

Get the Facts! 83
Conversations 85
Taking a Survey 85
Bar Graphs 90

Chapter 10

Topics for Discussion 93
Turtle Geometry: The Turtle's State 95
Grammar: Instructions and Inputs 97
Defining Procedures 99
Superprocedures and Subprocedures 101
Commands and Reporters 103
Procedures with Inputs 105
Recursion 107
Variables 109

Index 111

Introducing LogoWriter

In my vision, *the child programs the computer* and, in doing so, both acquires a sense of mastery over a piece of the most modern and powerful technology and establishes an intimate contact with some of the deepest ideas from science, from mathematics, and from the art of intellectual model building.

Seymour Papert, *Mindstorms,* 1980.*

In the six years since Papert wrote *Mindstorms,* the presence of computers in schools has been well-established. Two of the major uses of computers in elementary schools are application programs and programming. The most popular application programs are those concerned with word processing. A programming language is best exemplified by Logo, the computer language designed for children.

Because of a limited amount of time and available computers, many teachers find themselves in the position of having to choose between these two activities. On the one hand, word processing is immediately useful as a classroom tool. On the other hand, Logo gives students the power of programming, getting the computer to do what they want it to.

LogoWriter combines these two activities in an extension of Logo to include word processing. Logo graphics provides a concrete and meaningful context for experimenting with geometry concepts, and offers an easy entry into the world of programming. Word processing adds another dimension to this learning context, a dimension in which words and language are more meaningful and accessible.

The benefit of integrating Logo with word processing is not simply replacing two kinds of software with one. The combination of Logo and word processing produces a qualitative change in both computer environments.

Word processing enriches the environment of Logo programming activities. Text becomes an object that is interesting and can be easily manipulated and changed.

Word processing is demystified. It is no longer a "black box;" its features can be expanded and programmed just like graphics. Students can understand and control this word processing. They can personalize editing functions and create new ones.

LogoWriter is a learning tool for students. It provides a creative learning environment in which students can formulate ideas, construct systems, and create images. The most important objective in using LogoWriter is to develop learning skills and to explore different areas of knowledge.

S. Papert, *Mindstorms,* New York: Basic Books, 1980.

Chapter **1**

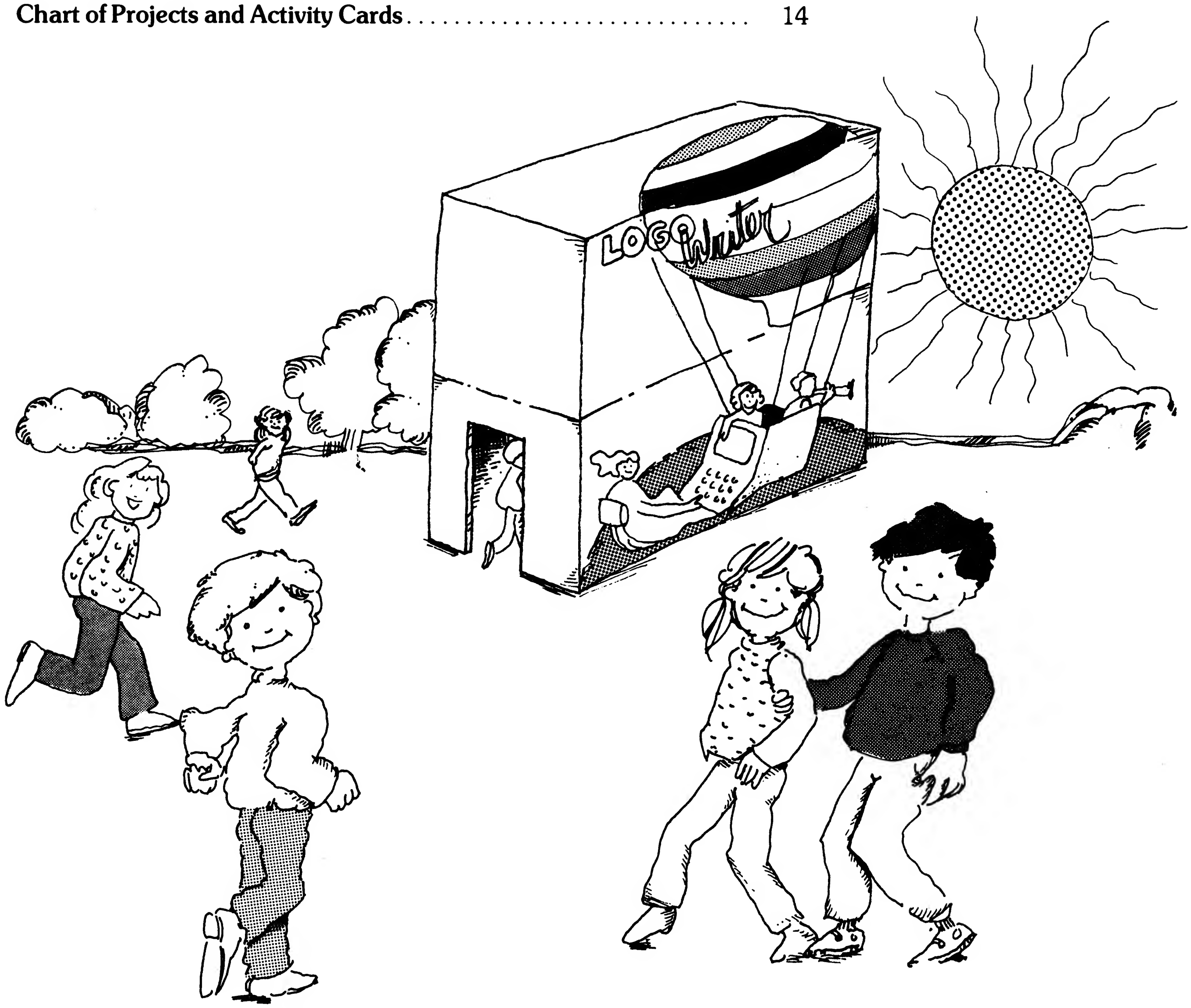
A Walk Through Your LogoWriter Kit

At a Glance 11

Using LogoWriter in Your Class.. 11

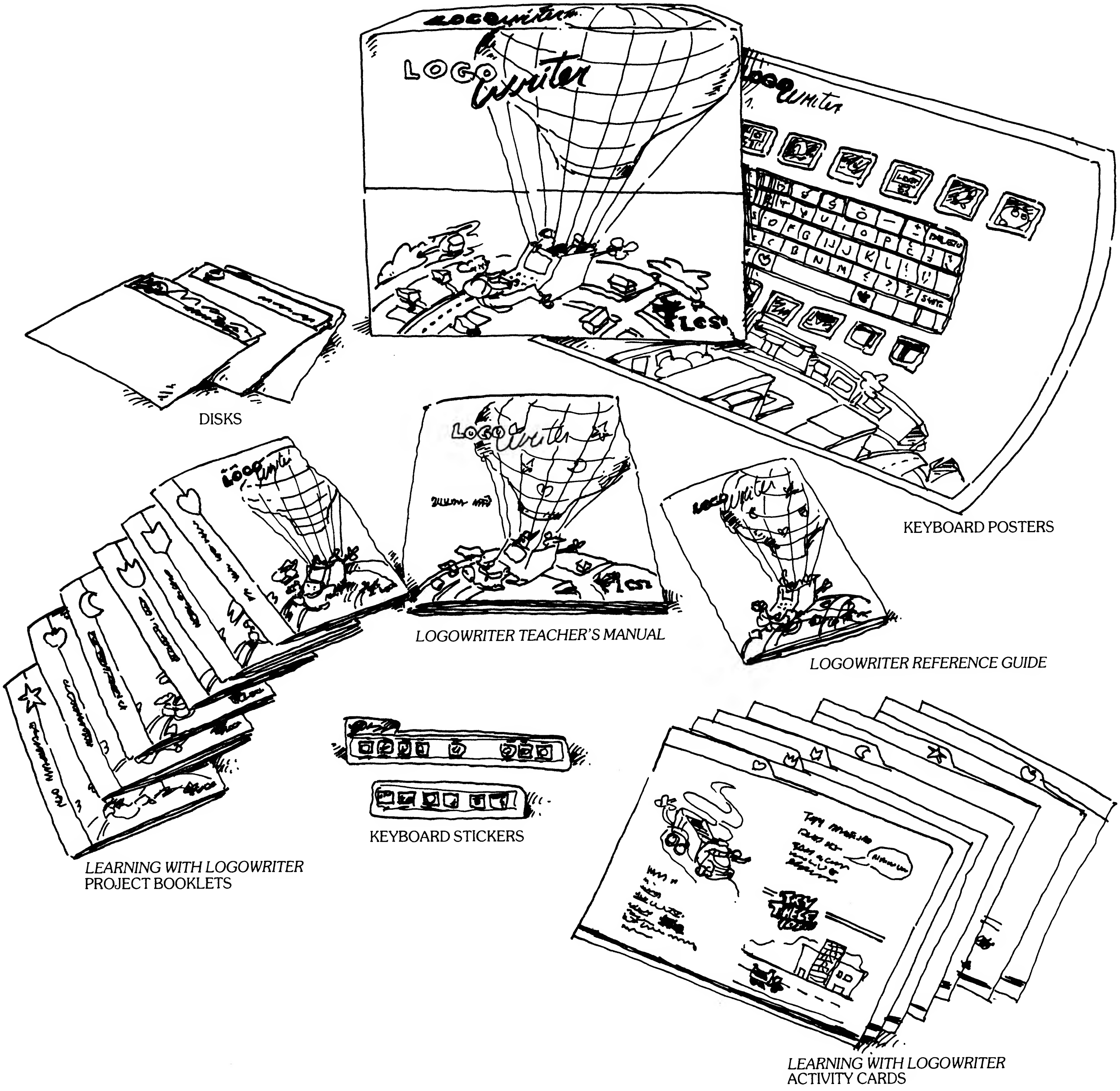
Getting Ready to Use LogoWriter With Students 13

Chart of Projects and Activity Cards.............................. 14



The LogoWriter kit is designed to facilitate your teaching requirements for LogoWriter in the lab or classroom. The rate at which your students progress through the material will depend on their level of expertise and the amount of computer time scheduled. The LogoWriter kit includes enough varied material for students to work on for at least a year.

The LogoWriter kit consists of:


LOGO Writer
DISKS
KEYBOARD POSTERS
LOGOWRITER TEACHER'S MANUAL
LOGOWRITER REFERENCE GUIDE
LEARNING WITH LOGOWRITER PROJECT BOOKLETS
KEYBOARD STICKERS
LEARNING WITH LOGOWRITER ACTIVITY CARDS

At a Glance

The **LogoWriter Program disk** is provided for your brand of computer. This disk contains the LogoWriter program and a selector for setting up printers with LogoWriter. It is important to make a backup of this disk and store the original in a safe place. Refer to the ***Supplement*** (included in the disk envelope) for information about setting up the LogoWriter disk, creating scrapbook (data) disks, and making copies. Refer to the ***LogoWriter Reference Guide*** for information about the different versions of LogoWriter.

This book, ***LogoWriter Teacher's Manual***, is your guide to using LogoWriter in the classroom. It explains the components of the kit. More specifically, it contains: a framework for using these materials to teach LogoWriter; a thorough discussion of each project; and an explanation of how and when to introduce LogoWriter concepts.

The ***Learning With LogoWriter*** **project booklets** are the basis of the students' materials. The booklets contain projects for the students to work through on the computer. These are the titles:

On Your Mark, Get Set, Go
Word Adventures
Words Come Alive!
You and Your Friends
News, Views, and Information
Get the Facts!

Each booklet is supplemented by a set of **activity cards.** The project booklets and activity cards are color-coded for easy access by students. The activity cards provide activities to introduce new ideas and practice techniques introduced in the projects.

The ***LogoWriter Reference Guide*** includes a listing of all the primitives (words built into LogoWriter's vocabulary) and keys in LogoWriter, with definitions and examples. The ***Supplement*** pages that fit into the ***Reference Guide*** binder contain instructions on setting up your LogoWriter disk(s) for classroom use.

Posters and **stickers** are provided, showing the important LogoWriter keys on each type of computer.

The kit contains one copy of each *Learning With LogoWriter* project booklet. To use these booklets effectively in your class, it is advisable to have one copy of each booklet *for each computer station.*

Using LogoWriter in Your Class

On Your Mark, Get Set, Go gives step-by-step instructions on getting started with LogoWriter and introduces the core features of LogoWriter. *Even those who have used other versions of Logo should read through this booklet to understand how LogoWriter works.*

The other project booklets present model projects for the students, to be used as a basis for their own projects. Each booklet contains two or three projects. Students learn how to use LogoWriter by doing the projects. It is important to this approach of "learning by doing" that students develop assigned projects in their own ways. Once students are comfortable using LogoWriter, they will be able to generate their own ideas for projects.

The project approach for learning LogoWriter is explained in Chapter 2 of this manual. This approach provides the framework for the LogoWriter materials in the kit. Read Chapter 2, *The Project Approach,* before you start using LogoWriter with students.

The *Learning With LogoWriter* project booklets are organized by themes and level of experience with LogoWriter. The themes provide broad ties to the school curriculum, making it easy to link LogoWriter with classroom activities. The booklets are sequenced from "no experience with LogoWriter" to "advanced." A chart of the booklets, their themes, and the chapter covering them in this manual, appears below.

Booklet	**Themes**	**Teacher's Manual**
♥ **On Your Mark, Get Set, Go**	Getting started.	Chapter 4
🌷 **Word Adventures**	Language arts, word processing techniques.	Chapter 5
☾ **Words Come Alive!**	Language and communication arts; animation and text.	Chapter 6
📖 **You and Your Friends**	Social studies, language arts, mathematics.	Chapter 7
★ **News, Views, and Information**	Social studies, language and communication arts.	Chapter 8
🍎 **Get the Facts!**	Mathematics, social studies.	Chapter 9

Working through each project introduces the students to new ideas, and, at the same time, reinforces old ones. The general rule of thumb is to progress through the project booklets in the above order. Use your own discretion in choosing the sequence. It depends in part on whether the projects are integrated into classroom activities. (See Chapter 3, *Organizing the Classroom.*)

Since each project introduces new LogoWriter ideas, working through all the booklets (supplemented by activity cards) provides experience in using the most important features of LogoWriter. If you change the order of the booklets (or the projects within them), check the concepts and skills covered in the projects that are being skipped. You can then introduce the new material yourself.

Chapters 4 through 9 of the *Teacher's Manual* provide the following information on each project:

Curriculum ties.
LogoWriter skills introduced or used.
Concepts to discuss.
Off-computer activities.
Difficulties students may encounter.
Ways to personalize the project.
Activity cards to be done after the project.

The conceptual information related to each project is presented in Chapter 10, *Topics for Discussion*. This chapter also provides guidelines indicating when each topic should be introduced.

Activity cards provide:

Practice in techniques and concepts.
Ideas of ways to extend a project.
Starting points for new projects.
"How to" techniques.

A list of the activity cards that supplement each project is found at the end of this chapter. The activity cards are sequenced to correspond to the same level of LogoWriter experience as the project. They are designed to develop a broad exposure to different LogoWriter skills, concepts and activities.

The gray section of the cards are *Reference Cards.* These are to be used for reference purposes. They provide students with simple explanations of LogoWriter techniques and primitives.

Students should work through each project in a project booklet. Activity cards supplementing the project should be made available when the project is completed. The order in which the cards for each project are completed is flexible, and may depend on student preference and availability of cards.

Getting Ready to Use LogoWriter with Students

Here is a quick guide:

Step 1: Try out LogoWriter.

Start up a copy of the LogoWriter disk and work through *On Your Mark, Get Set, Go.* This will give you experience with the important features of LogoWriter.

Step 2: Make copies of disks for your students.

It is advisable for each student or pair of students to have a scrapbook (data) disk or a copy of LogoWriter. Information about creating scrapbook disks and making copies of LogoWriter is found in the ***Supplement***, included in the LogoWriter disk envelope. Only site license holders are permitted to make copies of LogoWriter.

Step 3: Read Chapters 2 and 3 of this manual.

Chapter 2, *The Project Approach,* provides the conceptual framework of the kit including ideas on evaluation. Chapter 3, *Organizing the Classroom,* deals with the practical issues of classroom organization for computer use.

Step 4: Keep yourself up to date on the projects.

You need to do some preparation before the students start a *Learning With LogoWriter* project:

1. Work through the project on the computer.
2. Read the section of the chapter in this manual which covers the project.
3. Look through the activity cards that follow the project.

Chart of Projects and Activity Cards

Project Booklet	Color Activity Cards	Teacher's Manual
♥ **On Your Mark, Get Set, Go**	**Dark Green**	Chapter 4
	Drive a Truck Fill Your Drawings With Color! Building Blocks Tracks in the Sand Sunshine Flashing Sign Dandelions Label Your Pictures Bug Eyes What a Character! A Connect-the-Dots Puzzle	
🌷 **Word Adventures**	**Light Green**	Chapter 5
Crossword Puzzle		
	Letterhead Be My Valentine! Clock Star Tricks	
Adventure Story		
	Shape Your Stories Melody Score Sheet	
Secret Code		
	Invitation A Shortcut Give Yourself Credit One Key Typing One Key Graphic Tricks Save Time	
☾ **Words Come Alive!**	**Orange**	Chapter 6
Pun Fun		
	Tulip Garden Rocket Take Off Line Gobbler It's Snowing! Snowman On the Road Ferris Wheel	
Row, Row, Row Your Boat		
	One Square for All Sizes! Gift Box Helicopter Traffic Signs A Fill-in Procedure Draw Thick Lines Look at All Your Pages!	

Project Booklet	Color Activity Cards	Teacher's Manual
You and Your Friends	**Red**	Chapter 7
Walking Through Your Neighborhood	Climbing a Ladder A Turtle Eraser Sound Effects – IBM Only Sound Effects – Apple Only Table of Contents Caterpillar	
Dice 4	A Starry Sky Quick Change Big Titles	
Leaping Turtle	Spirals Bring Your Page to Life! A Rainbow!	
News, Views, and Information	**Purple**	Chapter 8
Lights, Camera, Action	A Talking Clown Address Book UFO! Shark!	
Traveling Through Time	Stop Rules: Spirals The Countdown Turtle Tag Turtle Squash Startup Your Scrapbook Change Your Mind?	
Get the Facts!	**Blue**	Chapter 9
Taking a Survey	Turtle Doodling Hopping Turtle Guess a Number	
Bar Graphs	Bar Charts Cartesian Grid 3-D Spiral In Beyond Polygons Digital Counter Bumper Turtles – Apple Only	

Reference Cards	Color
	Activity Cards
	Gray
	Turtle Graphics
	Four Turtles
	Using a Printer
	Word Processing – IBM Only
	Word Processing – Apple Only
	Search and Replace
	Defining Procedures
	Programming Control Keys
	Formatting Procedures
	The Shapes Page – IBM Only
	The Shapes Page – Apple Only
	LogoWriter Arithmetic
	The LogoWriter Keyboard – IBM
	The LogoWriter Keyboard – IBM PCjr
	The LogoWriter Keyboard – Apple
	Create Your Shapes – IBM Only
	Create Your Shapes – Apple Only
	Create Your Shapes – IBM Only
	Create Your Shapes – Apple Only
	Create Your Shapes – IBM Only
	Create Your Shapes – Apple Only
	Create Your Shapes – IBM Only
	Create Your Shapes – Apple Only
	Create Your Shapes – IBM Only
	Create Your Shapes – Apple Only

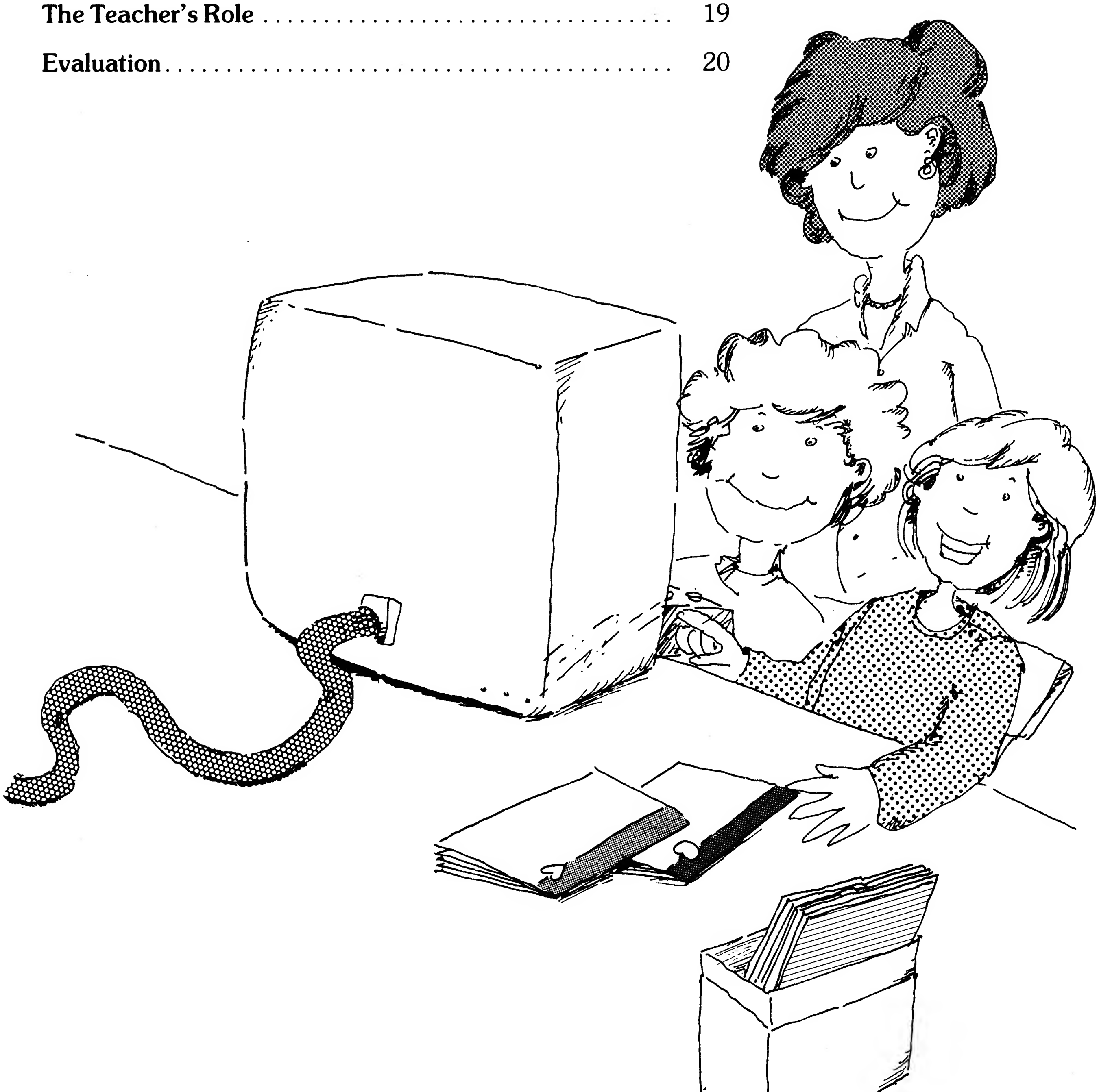
Chapter **2**

The Project Approach

Advantages of the Project Approach 18

The Teacher's Role 19

Evaluation 20



The framework of the LogoWriter kit is the project approach. Each project integrates different kinds of learning to construct a "product." Students learn new techniques and skills by realizing the project. Projects are open-ended; any project can be simplified or made more complex as it is being developed. Following are some examples of the projects:

A time line of historical events with descriptions and pictures of each event that is being recorded.

A map of the neighborhood that integrates text and animation.

A survey of television-watching, with a graph representing the results.

Advantages of the Project Approach

Students learn in a meaningful context.

The subjects of the projects are interesting to students. Consequently, they are highly motivated to learn the new skills that are necessary to complete the project. The students perceive the knowledge that they acquire as a means of developing the project. Thus, the knowledge acquired is applied and relevant.

Learning is individualized.

Students can work at their own pace and in their own style, modifying the project to suit their tastes and aptitudes. Consider the following examples. A number of students working on the same project could envision the goal differently: one student may diverge from the original goal; another student may compromise on the end-result when correcting errors. A student who has advanced quickly can make a project more complex; this student learns additional skills that were not included in the original project.

What is learned in an individualized or "personal" context is not likely to be forgotten.

Students can relate something new to something they already know.

The subject of each project is generally familiar to most students (for example, a crossword puzzle). This gives students a clear idea of the goal of the project and the steps to follow in order to attain it.

Students have experience integrating different kinds of knowledge.

Students must combine different skills and ideas (knowledge) in order to construct the project. For example, a student working on the map project must combine mathematical ideas (spacing and drawing the map), language arts skills (writing a description of each house), and programming ideas and techniques.

A personalized project is a reflection of the student's understanding.

If all students complete a project in exactly the same way, it is difficult to evaluate the progress of an individual student. In a personalized project, areas of understanding and confusion are more apparent.

The learner is in control of the process of learning.

This aspect may not be immediately apparent, and usually emerges after the students are familiar and confident with LogoWriter. The student determines the goal of a project, plans it, constructs the different parts, analyzes and corrects errors, and finally evaluates the product. Being in control of the process of learning is intrinsically motivating and gives students experience in taking responsibility for what they learn.

Students generate their own projects.

The goal of this approach is that students be able to generate their own projects rather than always relying on you or *Learning With LogoWriter* projects. Self-generated projects are even more meaningful and relevant. Because their own involvement is greater, students are more committed to finding solutions for their difficulties and are more "open" to learning new concepts.

The process involved in generating a project gives students the opportunity to develop initiative and creativity.

The *Learning With LogoWriter* materials provide students with experience developing many different types of projects. As a student works through the projects and corresponding activity cards, he or she can modify certain aspects of a given project. *The ultimate objective of the LogoWriter materials is to make students take initiative and become autonomous learners.*

The Teacher's Role

The materials in the LogoWriter kit are designed to create an open-ended learning environment. The open-endedness of this learning environment extends to your role. Ideally, you will be trying to create a learning environment for the students that encourages them to feel free to experiment. Don't expect to know all the answers, but help students by posing the right questions and keeping them focused on the important aspects of their work.

As in any teaching situation, the quality and timing of your intervention must be carefully thought out. Here are some points to consider:

Suggest projects that are within the students' abilities.

There is a range of projects provided; as well, many of the ideas in the activity cards can be developed into projects. If a student has an idea for a project that you know is too advanced, suggest modifications to reduce it to within his or her capabilities. The project can be developed further at a later time.

Provide the general framework for the project.

Give students the general outline of a project, so that they know what is involved. In some cases, you may decide to link aspects of the project to school and home activities.

Allow exploration.

Students need time to experiment with new ideas, even if this experimentation is not directed toward any clear goal. At first, it may be difficult to distinguish constructive exploration from simply playing around. Observation of the student's work will help you decide when the exploration is no longer productive.

Give ideas without always giving answers.

Help the students analyze their problem areas. Good judgement is required on your part. If a student is overwhelmed by a confusing situation, it is important to provide direct answers so that he or she can overcome the immediate problem. In other situations, it is often helpful to raise a question with a student or a group of students and let them try to figure out the answer.

Introduce new concepts and techniques when the students are ready to use them.

The timing of introduction of a new concept or technique can make a big difference in its impact on the students. Outlines for presentations of LogoWriter concepts as well as general guidelines for when they should be introduced in relation to the projects are given in Chapter 10, *Topics for Discussion*.

Review concepts and techniques that have been learned.

In completing a project, students often apply new concepts without understanding the process involved. Reviewing concepts or techniques after the students have used them in their work reinforces what has been learned.

Discuss the process of the project.

Encourage students to show the results of their projects to each other, and to discuss what was involved in doing a project. In this manner, they will realize that a project can be done in several different ways. Point out the advantages and disadvantages of each method of doing a project. Give students the opportunity to talk about the kinds of problems they encountered while doing the project, and the strategies they used for solving these problems.

Making students think about what they have done helps them develop an awareness of their own thinking. This kind of discussion also develops a respect for the thinking of others.

Evaluation

A student's work on LogoWriter projects should be assessed on an individual basis. This means looking at the student's progress without immediate comparison to peers. As in any subject area, it is important to observe the *process* of learning as well as the *results* of learning.

Your main tool for evaluating a student's progress is observation. When observing the process of learning in completing a project, it is useful to have a "checklist." The following are examples of items you might include on your checklist.

How the student uses activity cards.

Balance the quality of the student's work with quantity. Doing many activity cards does not necessarily mean the student is processing the learning.

Which activity cards were completed after the project?
Did the student try to complete many cards?
Did the student enhance each card with personal touches?

The student's comprehension of the project and cards.

Was the student able to follow the written instructions independently?

Did the student read the material carefully before asking questions?

Did the student read the instructions or simply copy the procedures?

Analysis of the student's errors.

Analysis of a student's errors can provide valuable insight into the areas in which the student needs extra help.

What kind of errors did the student make – technical and/or conceptual?

What are the reasons for the error – misunderstanding, carelessness, unawareness of a technique?

Does the student always have similar kinds of errors?

You should also consider the strategies the student adopts when confronted with an error.

Did the student attempt to correct the error?

Did the student try to find the solution independently?

Did the student stop working on the project when faced with an error?

Did the student's error lead to a change in the goal of the project?

Did the student use the error as an opportunity to explore something new?

Exploration and creativity.

Did the student add or change details of the project to make it personal?

Was the project the student's own idea?

How does the student generate project ideas: from a personal interest, from you, from other students, from a *Learning With LogoWriter* project or activity card?

Relationship to peers.

Observe the extent to which the student cooperates and shares ideas with others.

Does the student contribute ideas in a group situation?

Does the student impose ideas on the others?

Is the student open to suggestions given by the other students?

In evaluating the *results of learning* after completing a project, consider these points:

The result of the project.

Keep in mind that your definition of a finished project may differ from the student's. One student may consider a project finished when it is completed as written in the *Learning With LogoWriter* booklet. Another student may consider a project finished when a personal touch is added, or when the extensions suggested in the project are completed.

Does the project work as the student intended?

Does the result of the project correspond to the student's original goal?

Is the result of the project different from others the student has done?

What is the degree of programming complexity involved?

Time to complete the project.

If the student took a much longer time than others to complete the project, examine the reasons why. For example: the student did more than what was expected; the student had difficulties that had to be resolved. Likewise, if the student did the project in a short time, examine why. For example: the project was not challenging; the student compromised easily on errors; the student did not add anything extra to the project.

Applied techniques/skills.

Analyze the LogoWriter techniques the student used for the project.

What programming skills were used?
Did the student apply techniques used previously but not directly given in the project?
Did the student apply any completely new techniques?

While not directly related to your evaluation of the student's progress, it is important to consider the affective component of learning.

Is the student happy with the result?
What would the student have liked to do differently?
Does the student generally discuss the difficulties encountered while doing the project?
Does the student think that he or she learned something in doing the project?

Chapter **3**

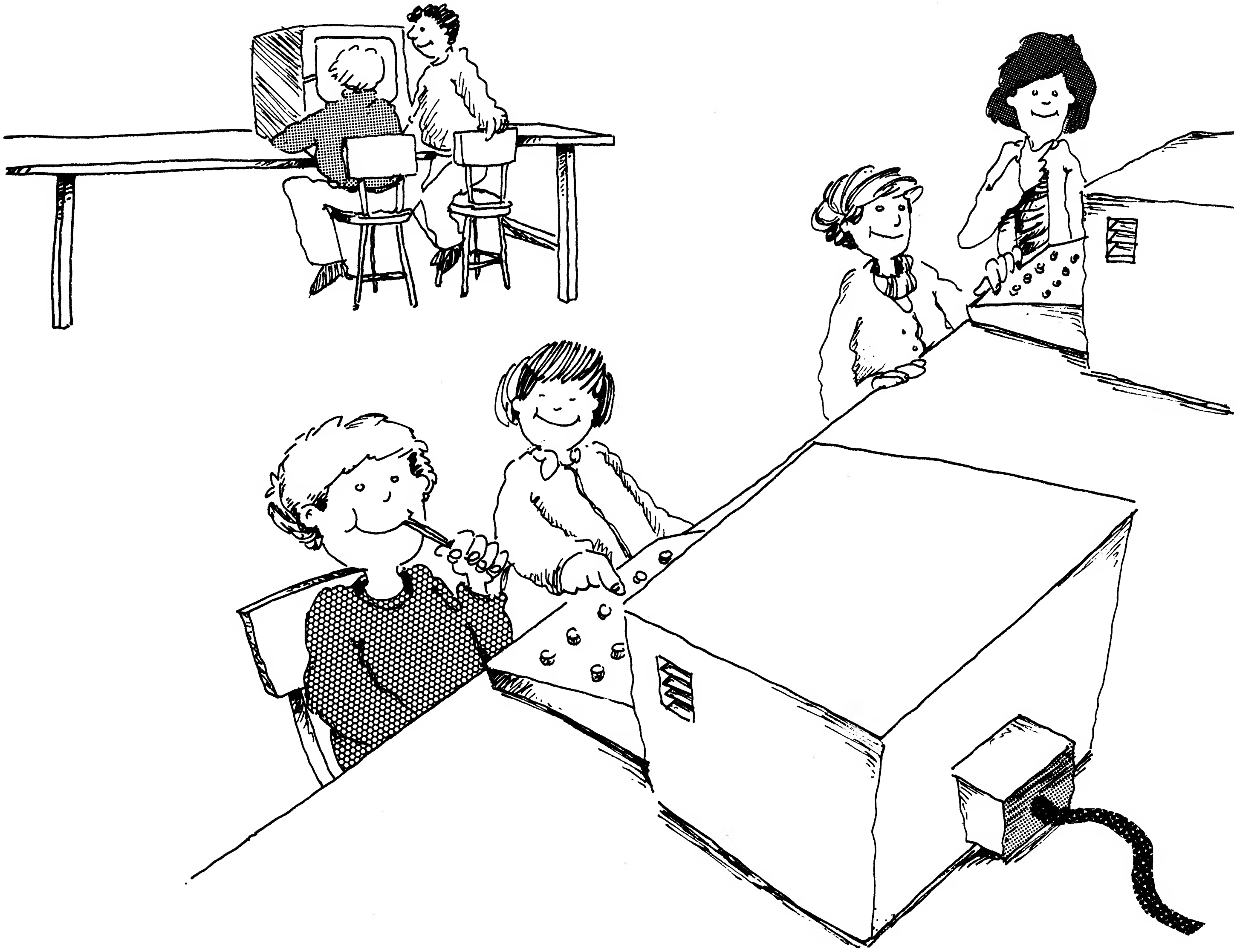
Organizing the Classroom

Computer Labs 24

Computers in the Classroom 25

Working in Pairs 26

Disks and Notebooks 26



Before using LogoWriter in your class, you must resolve some practical issues:

How to organize the use of LogoWriter in the class.

How to integrate the LogoWriter activities with the rest of the class curriculum.

Related to these issues are the number of computers available for your use, where they are located, your access to them, and your classroom organization.

In order to use the LogoWriter kit effectively, you will need:

Time when the whole class or group is together. This time should be used for introduction of new material, discussions, and sharing of students' work.

Time when the students can work at the computer, either individually or in pairs.

Informal discussion between students, and between you and the students, about the work they are doing with LogoWriter.

Computer Labs

Many schools have a computer lab which is available for scheduled periods.

Computer labs offer a number of advantages:

Equal access to the computers by all students.
All students working on computers at the same time.
Scheduling uninterrupted time for computer use.

The disadvantage is the isolation of the computer from regular classroom activities. In this situation, it is more difficult to make computer use spontaneous and connected to classroom activities. Making a link between computers and the curriculum is further complicated if the computer course is taught by a teacher other than the classroom teacher.

The nature of the LogoWriter materials facilitates a link to the curriculum. The projects in the *Learning With LogoWriter* booklets can be integrated into classroom activities. For example, a classroom teacher may want to introduce mapping or awareness of the neighborhood before the students create a map with LogoWriter. While a lab teacher may not have a general discussion on maps, the idea of a map project will probably link to a social studies project the students have done or will do at a later point. The familiarity of the subject matter will help make this project relevant to them. The extent of the link to classroom activities will depend on the communication between teachers, and between teachers and students.

A lab session can be divided into three phases:

- The first phase involves presenting a new idea (see Chapter 10, *Topics for Discussion*), doing off-computer activities to provide a general framework for the project, or reviewing and discussing previously presented topics. This phase is usually brief.
- The second phase is the "computer" time, when the students are working on their projects. During this time you can circulate among students, helping them with difficulties and observing their progress. In this informal teaching time, you may also be introducing new material to individuals or small groups of students. This phase constitutes the bulk of the lab session.

- The third phase of a session is the "wrap-up," when the students finish off whatever they are working on for the day. In general, this part of the computer session is short. However, the wrap-up segment can be longer when students finish a project and share their work with each other.

Computers in the Classroom

In some schools, one or a few computers are placed in each classroom where they are used throughout the school day. The advantage of this organization is the facility of integrating computer activities with other classroom activities. The disadvantage is fitting the computer time into a busy classroom schedule.

If you have computers in the classroom, you can find a way to use LogoWriter as a tool in diverse subject areas. For example, students can use LogoWriter to write their compositions for a language arts activity. As a geometry exercise, students can experiment with angles using turtle geometry. A discussion on graphs could include drawing graphs with LogoWriter.

This integration with the classroom curriculum exemplifies the objective of LogoWriter as a classroom tool. By using LogoWriter in everyday classroom activities, there is less emphasis on the technical aspects of programming. The goal becomes the content of the subject area rather than the technique being used.

Providing informal instruction to students working on the computer in the classroom may sometimes be difficult. There are a number of ways to encourage the students to be more autonomous in their work with LogoWriter:

Emphasize the use of the on-line tutorials for the beginning students. The tutorials introduce the beginner to the keyboard and turtle graphics commands.

Place the *Keyboard* Poster in clear view of the students. Remind students to refer to the poster and keyboard stickers.

Provide access to the activity cards that correspond to the projects students are working on.

Remind students to refer to Reference Cards (gray section of activity cards) for techniques and definitions of primitives.

Encourage interaction between students. Working in groups gives students their own "sounding board" for ideas and problem solving.

It will probably be necessary to schedule a few "time slots" per week for some formal introduction of new ideas, review, planning projects, and sharing work.

Try to ensure that all students have equal time on the computer.

Working in Pairs

Some teachers find that students make better use of limited computer time by working in pairs. This situation can be very rewarding for the students and for the teacher.

Students generate ideas together. They can discuss the direction of the project, their errors and strategies for correcting them. In this situation, they are also learning cooperation, sharing of ideas, and how to make practical decisions, such as who does the typing.

It is essential to allow flexibility in this form of cooperation. You may notice one student taking over a session because his or her ideas are developing more quickly than his or her partner's. Cooperation must provide for some independent work. However, if you notice that one student of a pair is constantly taking the lead, you might consider arranging a different partnership.

Disks and Notebooks

Each student or pair of students should have a LogoWriter disk for saving their work. It is also a good idea for each student to have a LogoWriter notebook or binder for keeping a record of his or her work. The notebooks can be used for:

Planning a project.
Making notes on difficulties and strategies for correcting errors.
Outlining important techniques.
Saving printed copies of LogoWriter work.
Recording completed activity cards.

Chapter **4**

On Your Mark, Get Set, Go

Introduction 28

WORKING THROUGH ON YOUR MARK, GET SET, GO 29

Communicating With the Computer 29

The Learn Pages 29

Projects
- Making a Birthday Card 29
- Writing a Fan Letter 30
- Street Scene 30

Practical Tips
- The Scrapbook Concept 32
- A LogoWriter Page 32
- Saving a Page 33
- Page Names 34
- Erasing a Page From the Scrapbook 34
- Erasing the Learn Pages 34
- The Help Page 34
- LogoWriter Messages 34

Introduction

On Your Mark, Get Set, Go covers the information about LogoWriter that students need in order to get started.

Unlike the other project booklets, *On Your Mark, Get Set, Go* is tutorial in style; students should be able to work through it fairly independently. There are three short projects at the end of the booklet: *Making a Birthday Card, Writing a Fan Letter,* and *Street Scene.* These projects provide ideas of "finished products" that can be created using LogoWriter, at the same time familiarizing students with the idea of a project.

Introduce the following concepts when the students have completed *On Your Mark, Get Set, Go.* (See Chapter 10, *Topics for Discussion.*)

Turtle Geometry: The Turtle's State

Grammar: Instructions and Inputs

On Your Mark, Get Set, Go introduces:

The Scrapbook concept.
The Contents page.
The concept of a LogoWriter page.
Turtle graphics.
Label text.
Regular text.
The Help page.

The following primitives are introduced in this booklet:

back, bk
cc
cg
fill
forward, fd
ht
left, lt
namepage, np
pd
pe
pu
repeat
rg
right, rt
setbg
setc
setsh
shade
shapes
st
stamp

The activity cards with dark green tabs go with *On Your Mark, Get Set, Go.* Cards that introduce a new primitive are marked with an asterisk. Direct students to these cards when they have completed the booklet.

Drive a Truck	Turtle graphics, repeating instructions, setting the turtle's shape.
Fill Your Drawings With Color!	Turtle graphics, filling shapes with color.
Building Blocks	Turtle graphics, shading areas.
Tracks in the Sand	Turtle graphics, setting the turtle's shape, stamping the turtle.
Sunshine	Turtle graphics, repeating instructions.
**Flashing Sign*	Turtle graphics, **label**, **wait**.
Dandelions	Turtle graphics, repeating instructions, setting the turtle's color and the background color.
Label Your Pictures	Label text, correcting errors in label text.
Bug Eyes	Turtle graphics, repeating instructions.
What a Character!	Turtle graphics, label text.
**A Connect-the-Dots Puzzle*	Label text, printing on a printer, **printscreen**.

Working Through On Your Mark, Get Set, Go

Note: It is important that the keyboard stickers are placed on the keyboard before students start working with LogoWriter. The LogoWriter key functions are summarized on the *Keyboard* poster and the *Keyboard* Reference Card (gray section of activity cards).

Throughout the project booklets and activity cards, the key functions for each computer are provided in the following order: the Apple computers; the IBM computers.

Communicating With the Computer

Most students will probably have heard the term computer programming. However, they may not have a clear idea of what programming is.

Programming means making the computer do what you want it to do. Since computers don't understand English, you have to use a language that both computers and people understand. LogoWriter is a programming language – it tells the computer what to do.

To get the computer to do something in LogoWriter, you need to use special words called *primitives*. Primitives are built into LogoWriter. They are used to give instructions that LogoWriter will carry out. Using primitives, you can do all kinds of things – draw pictures, write text, edit text, etc.

The Learn Pages

The Learn pages are tutorials which introduce:

The keyboard.

Graphics commands.

Commands with inputs.

They may be helpful for getting started.

Projects

The purpose of *Making a Birthday Card, Writing a Fan Letter,* and *Street Scene* is to illustrate fun and interesting projects the students can do with even a minimal mastery of LogoWriter. They are designed to provide ideas, rather than as models to be copied. Encourage students to adapt these ideas to create their own projects. Students may want to try a number of variations of each project.

Making a Birthday Card

PAGE 13

This project shows how to integrate text and graphics to make a birthday card. In the model provided, the turtle is set to a heart shape and stamped in different colors to decorate the card. Label text (a form of graphics) is used to write in the message. Students can experiment with other ways of decorating the card – either by stamping different shapes or using turtle graphics commands to draw pictures or designs.

The steps to erase a stamped shape or correct a letter in label text are outlined in the booklet. If a student wants to start the card again, the **rg** command can be given to reset the graphics to the way they are when a new page is displayed.

Students can print out their cards. Direct them to the *Using a Printer* Reference Card (gray section of activity cards) for information on printing.

- Confusion between **cg** and **rg**.

Cg stands for clear graphics. It clears all graphics (including label text) from the page. It does not affect turtle color, background color, or turtle shape.

Rg stands for reset graphics. **Rg** resets the graphics to the way they are when a new page is displayed. This means:

All graphics (including label text) are erased from the page.

The turtle is white and is wearing the "turtle" shape.

The background is black.

Writing a Fan Letter

PAGE 16

This project shows how to use the word processing features of LogoWriter to write a fan letter. Students should feel free to write any kind of letter, or some other kind of text, such as a story or a poem, if they prefer. It is important that students feel comfortable using the word processing capacity of LogoWriter.

Be sure to tell the students to hide the turtle (**ht**) before they go up to the page to type.

For information on how to print out the letter, direct students to the *Using a Printer* Reference Card.

- Confusion between "regular text" and label text.

Label text is a form of graphics. It is ideal for use when a few words of text are required, such as for labeling a picture. When the **label** keys are pressed, the label cursor appears in the lower right corner of the page. This cursor is moved using the **arrow** keys. The label cursor is easy to position on the page.

"Regular text" is generally used to write stories, poems or letters. To use regular text, press the **up** keys. The text cursor on the page will start flashing. Text always appears at the cursor position. Writing with regular text is just like using most other word processors: use the **Return** (**Enter**) key for a carriage return, the **Tab** key to indent, the **Delete (Backspace)** key to backspace, etc. LogoWriter provides a full range of text editing functions. Refer students to the *Keyboard* poster or the *Keyboard* Reference Card for information on the important keys. The advantages of regular text are: it is easy to correct and format; more text can be typed on a page.

Street Scene

PAGE 18

Street Scene involves drawing a row of buildings. This project is more involved than *Making a Birthday Card* and *Writing a Fan Letter.* Students can use turtle graphics commands to draw rectangles and squares for the buildings. Sample instructions are provided. The buildings can be filled or shaded. Details can be added by stamping shapes such as trucks or cars. Finally, label text can be used to "label" the buildings and the street. The end result of *Street Scene* can be quite impressive. An ambitious student may spend a lot of time perfecting and adding detail to the scene.

This project provides excellent experience using graphics commands and is a good lead-in to procedures. Students may return to the idea of drawing a street scene once they have learned how to define procedures.

- Errors when drawing with the turtle's pen.

  To erase a drawn line, show students how to put the pen's eraser down (with the **pe** command) and go back over the drawn line. For example:

```
fd 50
pe
bk 50
```

- Problems with **fill** or **shade.**

  Fill or **shade** "leaking."

  If there is a hole in the shape or area being filled or shaded, **fill** or **shade** will "leak" onto the rest of the page.

  Fill not filling:

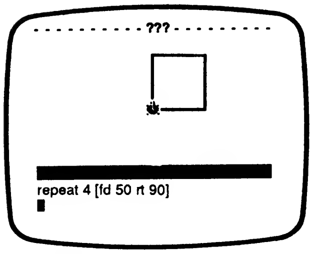
  The **pd** command must be given before the **fill** command.

  If the turtle is over a drawn line when the **fill** or **shade** command is given, only that line will be filled or shaded. It will appear as if nothing has happened.

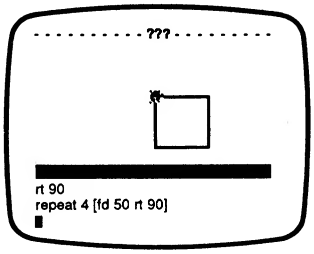
- Difficulty positioning the turtle to draw the buildings.

  Students generally learn a lot about positioning the turtle and the effect of turtle position and heading on graphics just by experimentation. If you notice that individual students are confused about the placement of the squares or rectangles in relation to turtle position or heading, you can take this opportunity to discuss these concepts.

  1. The direction the turtle is heading (facing) will affect the direction and orientation of the square or rectangle. Show the student examples such as the following:

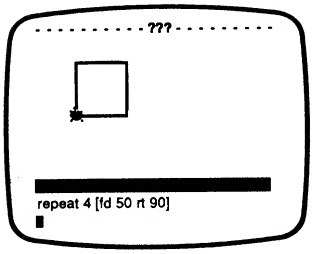


```
repeat 4 [fd 50 rt 90]
```

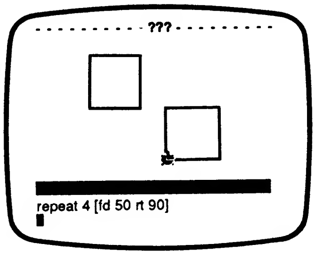


```
rt 90
repeat 4 [fd 50 rt 90]
```

  2. The position of the turtle on the page will affect where the square or rectangle is drawn in relation to other shapes (buildings) on the screen.



```
repeat 4 [fd 50 rt 90]
```

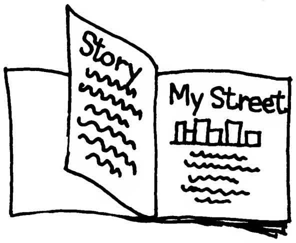


```
repeat 4 [fd 50 rt 90]
```

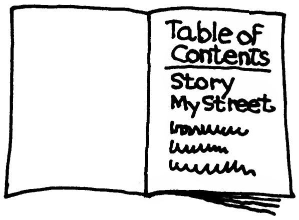
The Scrapbook Concept

The LogoWriter disk contains the LogoWriter program and a "scrapbook." The scrapbook is simply the LogoWriter files on the disk. Each file on the disk is called a "page."

It should be easy for students to relate to the scrapbook analogy. Encourage them to think of their LogoWriter scrapbook as being similar to a paper scrapbook. It comes with some pages filled in; other "blank" (empty) pages can be used for their own work. Every time they have finished working on a page, they can turn to a new page. To look at work done earlier, the student can go back to that page.



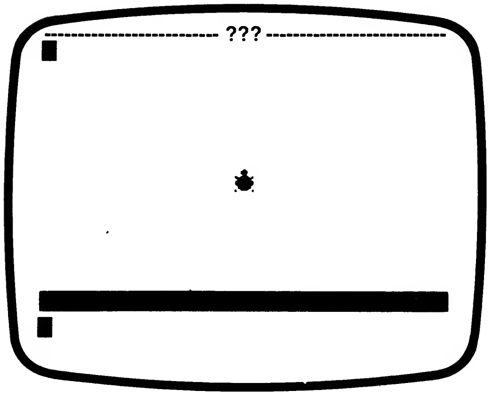
The Contents page is like a "Table of Contents." It shows the names of the pages in the scrapbook.



The first time the LogoWriter disk is used, the scrapbook consists of the following pages: the Shapes page, the Learn pages (tutorials), the Help page. Any page that is listed in Contents can be chosen.

A LogoWriter Page

A LogoWriter page looks like this:



This is the page.

This is the command center.

The command center is where LogoWriter instructions are typed. The page is where graphics are drawn or text is written. *The command center is not part of the page.*

Saving a Page

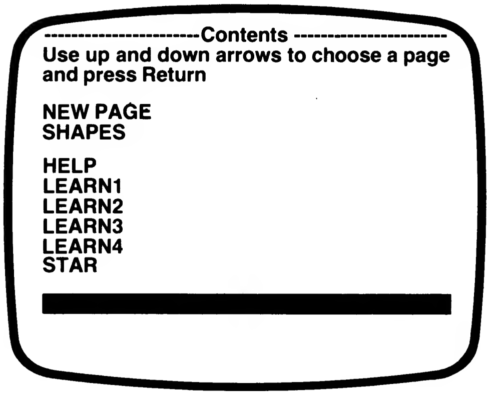
To save a page in the scrapbook:

1. Name the page. For example:

```
namepage "star
```

2. Press **Escape**.

When **Escape** is pressed, LogoWriter goes back to the Contents page. The page named star will be listed in Contents.



```
----------------------Contents ----------------------
Use up and down arrows to choose a page
and press Return

NEW PAGE
SHAPES

HELP
LEARN1
LEARN2
LEARN3
LEARN4
STAR
```

Pressing Escape automatically saves the page. It is equivalent to the **save** command in other versions of Logo.

If **Escape** is pressed before the page is named, LogoWriter prints a message:

Please name this page

A page can be "renamed." This feature is useful if a student wants to have two "copies" of one page (two identical pages):

1. Name the page. For example:

```
namepage "birthday
```

2. Press **Escape**.
3. Choose the birthday page from Contents.
4. Name the page again. Choose a different name. For example:

```
namepage "card
```

5. Press **Escape**.

The birthday page and the card page both appear in Contents.

Page Names

When students begin using LogoWriter, they typically use their own names, their friends' names, or even silly names for their pages. Encourage them to use meaningful names for their pages. Suggest that they use a name that describes what is on the page. This will help them find the page when they look at the page names in Contents. It will also help you identify their work if you want to look at it after a computer session. In each of the projects in *On Your Mark, Get Set, Go* a page name is suggested.

Note: LogoWriter has certain restrictions on page names. Page names must conform to the computer's file name conventions; among these are the following:

- Spaces cannot be used in page names
- On Apple computers, a number can't be the first character in a page name (i.e.: 1house, 2square)
- On IBM computers, a page name can't be longer than 8 characters; on Apple computers a name can't be longer than 12 characters (version 1.1) or 15 characters (version 2.0); on a COMMODORE 64 a page name can't be longer than 12 characters.

Erasing a Page From the Scrapbook

To erase a page from the scrapbook:

1. Press **Escape** to go to the Contents page.
2. Use the **down arrow** key to place the cursor on the name of the page to be erased.
3. Press the **erase to end of line** keys.

Tell students this is like ripping a page out of their scrapbook. This process is equivalent to the **erasefile** or **erf** command in other versions of Logo.

Note: In version 2.0, pages can be locked to protect them against accidental erasure. A locked page must be unlocked before it can be erased. See **lock** and **unlock**, in the ***Reference Guide.***

Erasing the Learn Pages

When the tutorials have been completed, they can be erased from the scrapbook to give more space on the disk for other pages. See the instructions *Erasing a Page From the Scrapbook* above.

The Help Page

The Help page contains information about the page commands, clearing the page, and the colors and shapes. There are two ways to get the Help page:

- Choose it from Contents.
- If you are on a page, press the **help** keys.

Press **Escape** to return to Contents or to the page you were on.

LogoWriter Messages

When you type a line that LogoWriter doesn't understand, it prints a message in the command center. The message describes the problem. For more information on these messages, see Appendix 5, *LogoWriter Messages,* in the *LogoWriter Reference Guide.*

Chapter **5**

Word Adventures

Introduction 36

CROSSWORD PUZZLE 37
Off-computer Activity 37
A Blackboard Version of a Crossword Puzzle 37

Working Through the Project on the Computer 38
Drawing a Grid With Procedures 38
Filling the Center Square 40
Numbering the Boxes 40
Typing the Clues 40

Going Further 40

ADVENTURE STORY 41
Off-computer Activities 41
The Elements of a Pick-Your-Own-Ending Adventure Story . . . 41
Planning the Computer Version of *Adventure Story* 42

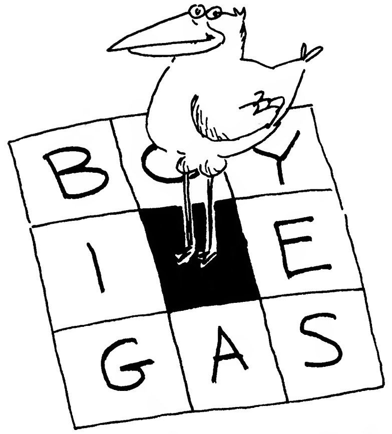
Working Through the Project on the Computer 42
Writing the Story 42
Changing Pages 42

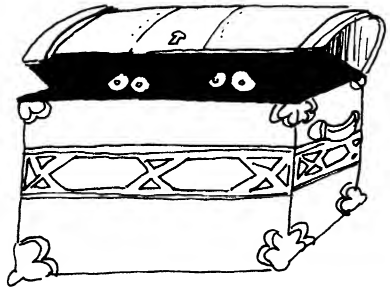
Going Further 43

SECRET CODE 43
Off-computer Activities 44
Coding Messages 44
A Pencil and Paper Version of *Secret Code* 44
LogoWriter Arithmetic Reporters 44

Working Through the Project on the Computer 45
Using Text Editing Primitives 45
Procedures to Code and Decode the Message 45

Going Further 46





Introduction

The projects in this booklet are specifically related to language arts skills. *Crossword Puzzle* integrates two skills – manipulating words and using turtle graphics. *Adventure Story* involves writing a story with a plot that branches. In *Secret Code*, word processing commands are used to develop "coding" and "decoding" techniques.

In each project, the student is shown how to define a procedure for each part or segment of the project, and then *run* the whole project under procedure control. Use of procedures and the structure of superprocedures and subprocedures are basic to LogoWriter programming. These are concepts that you will have to review and reinforce often while the students are working through the projects. *Secret Code* is the first project to use a reporter.

Introduce the following concepts in conjunction with these projects. (See Chapter 10, *Topics for Discussion.*)

Defining Procedures – Before *Crossword Puzzle*
Superprocedures and Subprocedures – After *Crossword Puzzle*
Commands and Reporters – Before *Secret Code*

The activity cards with light green tabs go with the projects in *Word Adventures*. The following chart indicates the correspondence of cards to projects. Cards that introduce a new concept or primitive are marked with an asterisk.

Crossword Puzzle

Letterhead	Label text, printing on a printer, stamping shapes.
Be My Valentine!	Regular text, stamping shapes.
Clock	Label text, using the **px** pen.
Star Tricks	Defining procedures, turtle graphics.

Adventure Story

Shape Your Stories	Regular text, using the **turtle-move** keys, turtle graphics.
**Melody*	Procedure definition, **tone**.
**Score Sheet*	Regular text, using the **tab** key.

Secret Code

**Invitation*	Regular text, **replace**.
**A Shortcut*	Using the **copy** and **paste** keys to define procedures.
**Give Yourself Credit*	Word processing commands (**cu**, **cd**), defining procedures, **print**.
**One Key Typing*	Programming keys (**when** command).
**One Key Graphic Tricks*	Programming keys (**when** command).
**Save Time*	Word processing commands, programming keys (**when** command).

Crossword Puzzle

PAGE
3

This project involves planning and drawing a crossword puzzle. Turtle graphics commands are used to draw the grid. Label text is used to fill in the numbers. Either label text or regular text can be used to write the clues.

Crossword Puzzle is an excellent exercise in vocabulary enrichment. If your students are studying a foreign language, they might want to create a crossword puzzle in this language.

Crossword Puzzle provides experience in:

Defining procedures.
Superprocedures and subprocedures.
Label text.

The following primitive is introduced in this project:

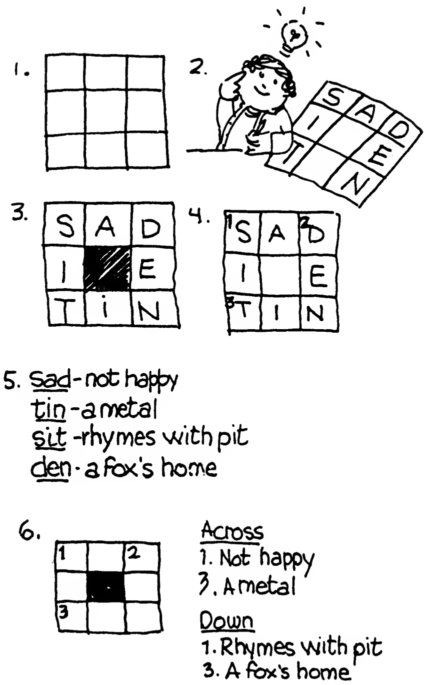
printscreen

Off-computer Activity

While most students are familiar with crossword puzzles, designing a crossword puzzle is a more difficult task than filling one in. You may want to give the students a few crossword puzzles to do on paper before they embark on this project.

A Blackboard Version of a Crossword Puzzle

Tap on the students' experience with crossword puzzles to review the steps for creating one. Apply each of these steps to a "blackboard" version of a crossword puzzle.



1. Draw the grid for the crossword puzzle.

Use a three-by-three box grid, like the one illustrated in the project booklet.

2. Fill in the words.

Let the students think of the words for the puzzle.

3. Fill in any boxes on the grid that are not being used for a letter.

4. Number the boxes.

See if the students can think of the convention for numbering the boxes: The first letter of every word must be numbered, but no box can have more than one number. The boxes are numbered from left to right and top to bottom. Let the students figure out how the numbers go in the puzzle you are developing.

5. Think of a "clue" for each word.

Clues must be concise. See what kinds of clues the students can come up with. They might suggest definitions, rhymes, synonyms, antonyms, etc.

6. Write in the clues.

The clues should be classified as <u>Across</u> and <u>Down</u>.

Working Through the Project on the Computer

Students should think of their own words for the crossword puzzle, rather than simply copying the example given in the project booklet or the one on the blackboard. Probably the most challenging task for them will be thinking of the words and the clues. Emphasize collaboration with friends.

Drawing a Grid With Procedures

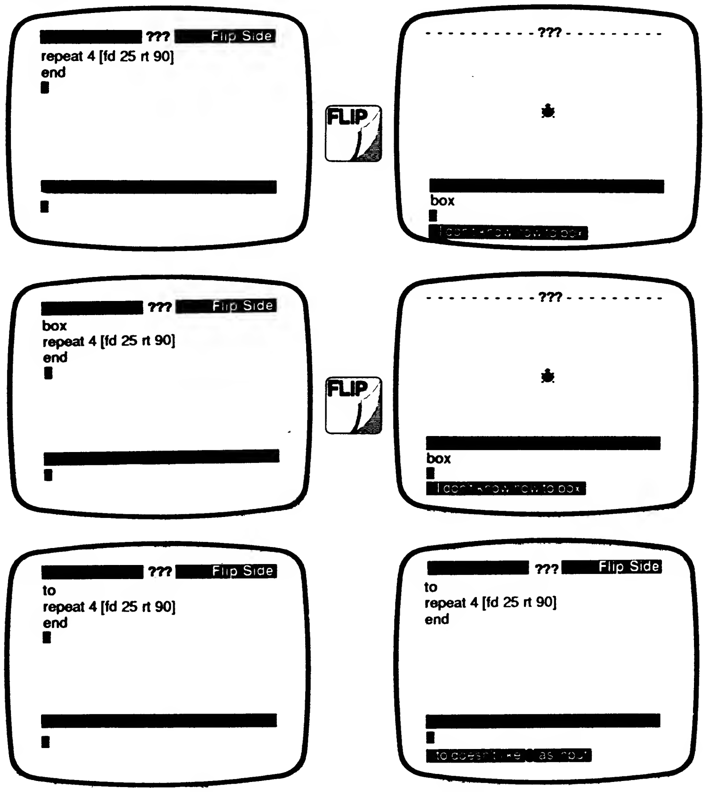
Sample procedures to draw the grid are provided. There are numerous ways to draw a grid. Encourage students to find a different method of drawing the grid.

Remind students to leave a blank line between each procedure definition. For example:

```
to box
repeat 4 [fd 25 rt 90]
end

to row
repeat 3 [box fd 25]
end

to______
.
.
.
```

- Forgetting to name a procedure.

  A procedure definition must begin with **to** and the name of the procedure. If the title line is omitted, LogoWriter will not recognize the instructions as a procedure.



```
repeat 4 [fd 25 rt 90]
end
```

```
box
I don't know how to box
```

```
box
repeat 4 [fd 25 rt 90]
end
```

```
box
I don't know how to box
```

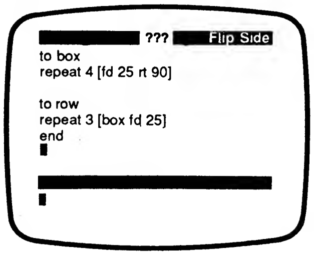
```
to
repeat 4 [fd 25 rt 90]
end
```

```
to
repeat 4 [fd 25 rt 90]
end

to doesn't like [] as input
```

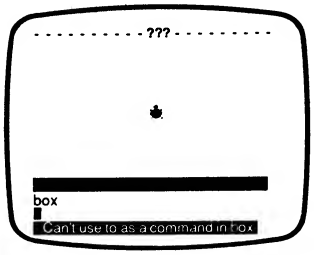
In the latter case, LogoWriter prints its message in the command center. To get back to the page and make the necessary correction(s) in the procedure definition, press the **up** keys.

- Forgetting **end**.

  The last line of a procedure is **end**. **End** tells LogoWriter that the procedure definition is complete. It separates one procedure from the next.



```
??? Flip Side
to box
repeat 4 [fd 25 rt 90]

to row
repeat 3 [box fd 25]
end
```

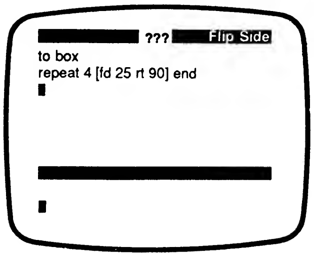


```
- - - - - - - - - - ??? - - - - - - - - -

box
Can't use to as a command in box
```

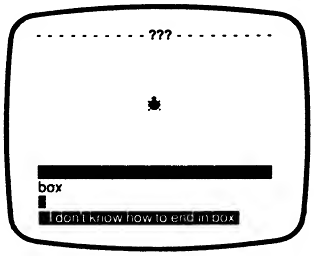
LogoWriter thinks the line **to row** is part of the procedure **box**, and tries to run this line. It realizes that **to** is a *special word* that cannot be used as a command.

- Putting **end** on the same line as another LogoWriter instruction.

  End must be on a line by itself. If **end** is on the same line as another LogoWriter instruction, LogoWriter will try to run it as an instruction. **End** is a *special word*.



```
??? Flip Side
to box
repeat 4 [fd 25 rt 90] end
```



```
- - - - - - - - - - ??? - - - - - - - - -

box
I don't know how to end in box
```

- Difficulty getting back to the front side of the page.

  The **flip** keys are used to flip from the front of the page to the flip side, and from the flip side to the front of the page.

- Trying to run a procedure from the flip side of the page.

  If a student tries to run a procedure from the flip side of the page, nothing will happen. LogoWriter ignores any text on the flip side of the page that doesn't start with **to** and end with **end**.

  Generally, when the procedure definition is complete, the **flip** keys are pressed to return to the command center and the front of the page. However, the student could press the **down** keys and go from the flip side directly to the command center. In this case, the procedure can be run from the command center. If the procedure contains graphics commands, LogoWriter automatically flips to the front of the page.

Filling the Center Square

In the project, **fill** is used to fill in the center square on the grid. There are other means of filling in this square. Examples: label text, drawn lines, a stamped shape.

- **Fill** "leaking."

  If the shape is not closed, **fill** will "leak" onto the rest of the page. Make sure the shape is closed before giving the **fill** command.

- **Fill** not filling.

  The **pd** command must be given before the **fill** command.

  Fill fills in the pen color. If the pen color is the same as the background color, it will appear as if nothing happened.

  If the turtle is over a drawn line when the **fill** command is given, only that line will be filled. It will appear as if nothing has happened.

Numbering the Boxes

- Forgetting how to return to the command center from label text.

  The **Escape** key must be pressed to return to the command center.

- Making a typing error in label text.

  To erase a typing error in label text, the same character must be retyped over the incorrect character. For example:

  Atross

  To erase the **t**, use the **arrow** keys to place the label cursor over the **t** and then press the **t** key. Press the ← key, and type a **c** in the blank space.

Typing the Clues

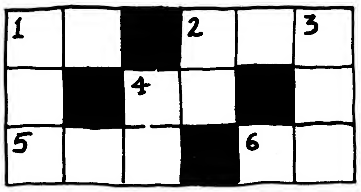
Students can write the clues with label text or regular text.

Going Further

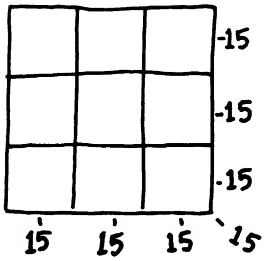
- If there is a printer in the class, students can print out their puzzles. Direct them to the *Using a Printer* Reference Card (gray section of activity cards) for further information on printing. You could collect all the crossword puzzles and distribute them to the class or arrange an exchange of crossword puzzles with another class.

Two follow-up activities are suggested in the project:

- Two grids can be put together to make a larger, more complex crossword puzzle.



- The grid can be used to make a crossnumber puzzle.



Rules:
All the rows must add up to the same number.

Only use numbers less than 10.

Use each number only once.

Adventure Story

PAGE 13

This project involves planning and writing a "pick-your-own-ending" adventure story. The story is typed using regular text, providing practice with the text editing features of LogoWriter as well as the more general language arts skills involved in writing a story.

This project can be integrated into a more global language arts program on adventure stories or pick-your-own-ending adventure stories.

Adventure Story provides experience in:

Using LogoWriter's word processing and text editing features.
Defining procedures.
Changing pages under procedure control.

The following primitive is introduced in this project:

getpage

Off-computer Activities

If your students are not familiar with pick-your-own-ending stories, you might want to give them one or two examples before they begin this project.

The Elements of a Pick-Your-Own-Ending Adventure Story

Ask the students to specify the important elements of a pick-your-own-ending adventure story. The following points are important:

The story must have a cohesive plot.

There must be an element of action and suspense in the story if it is to qualify as an adventure story.

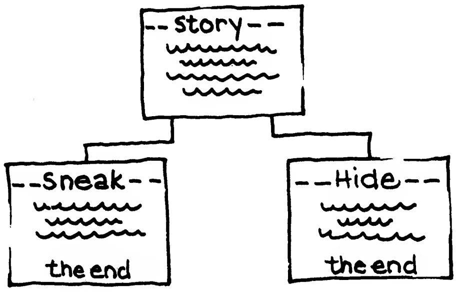
The plot must be developed to a point where the reader is given a choice of what action to take. This branching should occur at some "critical" stage in the story. The ending of the story will depend on the choice the reader makes.

Planning the Computer Version of Adventure Story

See if the students can think of any advantages of writing a pick-your-own-ending adventure story on the computer. They might mention the ease of typing and editing the text, the effect of having each part of the story on a different LogoWriter page, or the use of LogoWriter graphics to illustrate the story. Ask them how they would write a pick-your-own-ending adventure story on the computer. The steps are as follows:

1. Write the beginning of the story on one page. The plot must be developed to a point where the reader is given a choice of two endings.

2. Write each of the endings on a separate page. Each branch ends the story. You can diagram this branching process on the blackboard.



3. Go back to the first page. At the point where the story branches, the author will have to inform the reader of the two choices, and how to choose each of the endings.

Working Through the Project on the Computer

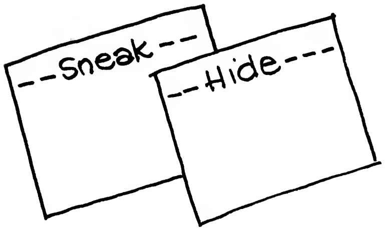
The most challenging task for the students will be thinking of the plot for their stories and developing it into a choice of endings. They can collaborate with classmates for ideas. Alternatively, you may want to give them an example of a first paragraph or opening sentence that they can develop.

Example: Three friends decided to explore an old shack. When they got to the shack, Nick said, "Maybe somebody lives here."
"No," said Kate. "Go in and you'll see it's empty."

Writing the Story

Students should have little difficulty typing the story. Remind them to refer to their keyboard stickers and the *Keyboard* poster on the editing keys.

Encourage the students to name their pages appropriately. They might be able to add an exciting touch to their stories with the names of the pages. Examples: Howls, Ghost, Shadow, Haunted. You may need to remind the students how to use the **up** and **down** keys to go from the command center to the page and back again.

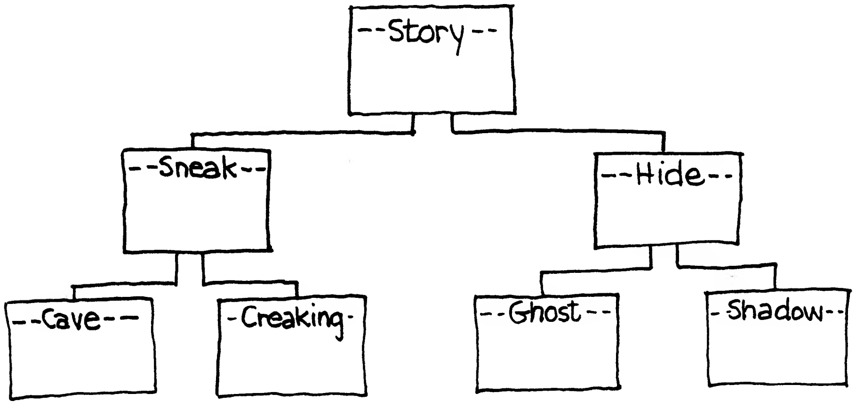


Changing Pages

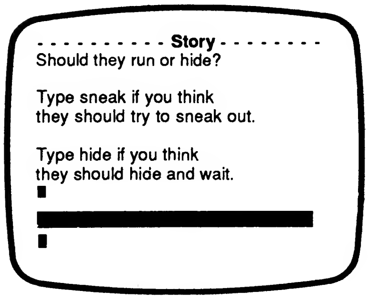
The project shows how to define procedures to change pages. The **sneak** and **hide** procedures use the **getpage** command to "get" a page. To avoid confusion, it is best to name the procedure the same as the page name, as in the sample procedures in the project.

Going Further

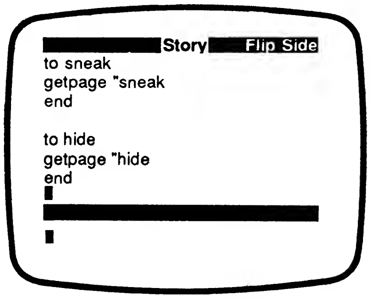
Write an adventure story that has more than one branch.



The procedure to change pages must be on the flip side of the page containing the choice. For example:



```
- - - - - - - - - - Story - - - - - - - -
Should they run or hide?

Type sneak if you think
they should try to sneak out.

Type hide if you think
they should hide and wait.
```



```
Story   Flip Side
to sneak
getpage "sneak
end

to hide
getpage "hide
end
```

Secret Code

PAGE
18

Secret Code involves defining procedures which use text editing commands to code and decode messages. Students typically find coding and decoding messages a fascinating process.

This project gives students a different perspective on word processing techniques. The coding and decoding techniques used involve inserting and deleting characters in the text. Students will be familiar with how to perform these functions using the text editing keys. *Secret Code* provides a fun example of the power and advantages of programming these text editing functions.

Secret Code provides experience in:

LogoWriter's word processing and text editing features.
Using text editing primitives.
Defining procedures.

The following primitives are introduced in the project:

cf
ct
delete
insert
show
textlen
top

Off-computer Activities

This project is the students' first introduction to text editing primitives. For this reason, it is a little more difficult to follow than *Crossword Puzzle* and *Adventure Story*. **Take the time to look over the project carefully yourself before you introduce it to the students.**

Coding Messages

As a general introduction to the project, you can discuss what a secret code is: a system used for secrecy of communication. Ask the students if they can think of different kinds of coding systems. Morse code is an example they may be familiar with. In morse code, dots, dashes and spaces are used to symbolize the content of a message. Can the students think of other ways to put a message into a secret code?

A Pencil and Paper Version of Secret Code

Introduce the method of coding used in the project – choosing a letter and adding this letter after every letter in the text.

Zachary
We will all meet in the . . .

Zaaacahaaaraya
Waea waialala aalala maeaeata iana tahaea

Each student could use this method to write a pencil and paper "secret" message to another student. The students can exchange coded messages and try to "decode" the message received. This exercise will clarify the techniques involved in coding and decoding a message, thus making the process easier to follow in the project.

LogoWriter Arithmetic Reporters

You may want to take a few minutes to explain the LogoWriter arithmetic reporters, as they are useful in the day-to-day use of LogoWriter. The decoding procedure in the project uses the division reporter. Be sure that you have discussed reporters with the class before introducing the arithmetic reporters. See *Commands and Reporters* in Chapter 10, *Topics for Discussion*.

LogoWriter can be used as a calculator – to perform simple arithmetic operations. LogoWriter can add, subtract, multiply and divide. Show the students the symbols for these operations. They can experiment with them on the computer.

	Symbol	Example
addition	+	**show 5 + 5**
subtraction	–	**show 10 – 6**
multiplication	*	**show 3 * 9**
division	/	**show 21 / 7**

Refer students to the *LogoWriter Arithmetic* Reference Card for more information.

Working Through the Project on the Computer

The procedures for coding and decoding a message are provided in the project. It is important that the students understand the *process* involved in the procedures so that they can use them independently at a later time.

Using Text Editing Primitives

The project shows how to use text editing primitives to code the student's name. The **insert** command is used to insert letters on the page. The **textlen** reporter is used to calculate how many times to repeat the coding instructions.

- Difficulty understanding **textlen**.

  If students are confused by **textlen**, show them this example. Type:

  ```
  show textlen
  ```

  Textlen will "report" the number of characters on the page to **show**. This number will be shown in the command center. In the instruction

  ```
  repeat textlen [cf insert "a]
  ```

  textlen reports the number of characters on the page to **repeat**. **Repeat** uses this number as its input and repeats the instructions in the brackets this number of times. It repeats the instructions once for every character on the page.

  Show the students a concrete example. If the name Gaby is the only text on the page, **textlen** will report the number 4 to **repeat**. The instructions **cf insert "a** will be repeated four times.

  ```
  Gaaabaya
  ```

- Confusion about the difference between the **print** and **insert** commands.

  Students who have worked with other versions of Logo may be confused about the difference between **print** and **insert**. Both commands print their inputs on the page. **Print** adds a carriage return after the input; **insert** does not follow its input with a carriage return.

Procedures to Code and Decode the Message

The **code** and **decode** procedures are given in the project. Encourage students to experiment with these procedures, using them to code and decode different messages. What is the effect of changing the letter that is inserted in the **code** procedure? Do some letters make the message more difficult to read than others?

"Bugs" encountered will probably be related to typing errors or to modifying the procedures provided.

- Difficulty understanding why **textlen** is divided by 2 in the **decode** procedure.

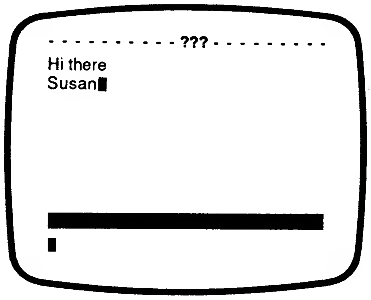
  To decode the message, it is necessary to erase every second letter, or half of the characters on the page. If the students have trouble understanding this, show them a concrete example:

  Gaaabaya

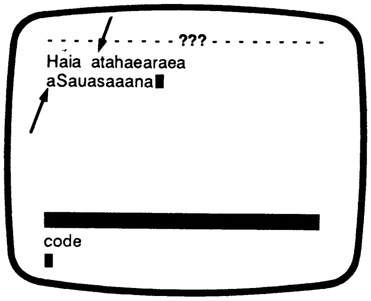
There are 8 letters in the word. To decode it, it is necessary to erase every second letter (a total of four letters).

- Difficulty understanding why the **code** procedure inserts an **a** after a space or a carriage return.

  In LogoWriter, the space and the carriage return are characters. For example:



```
- - - - - - - - - - ??? - - - - - - - - -
Hi there
Susan▮

▮
```

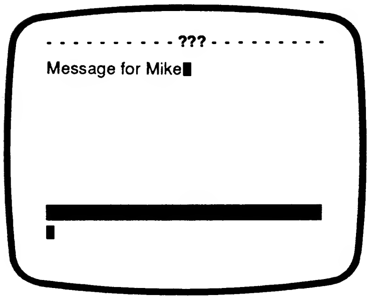


```
- - - - - - - - - - ??? - - - - - - - - -
Haia atahaearaea
aSauasaaana▮

code
▮
```

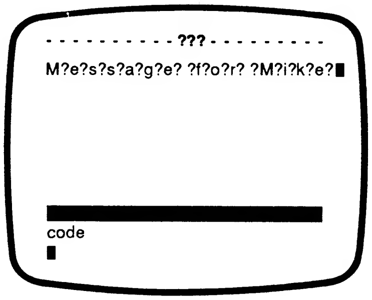
Going Further

See if the students can think of different techniques for coding and decoding. Which is the most difficult to read? You can suggest the following ideas:

- Inserting a number or punctuation mark after every letter.

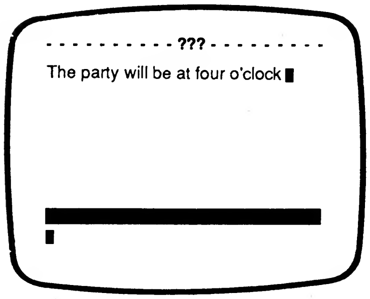


```
- - - - - - - - - - ??? - - - - - - - - -
Message for Mike▮

▮
```

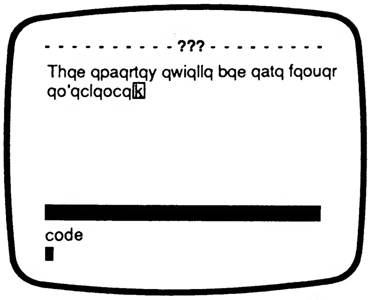


```
- - - - - - - - - - ??? - - - - - - - - -
M?e?s?s?a?g?e? ?f?o?r? ?M?i?k?e?▮

code
▮
```

- Inserting a letter after every second letter in the message.



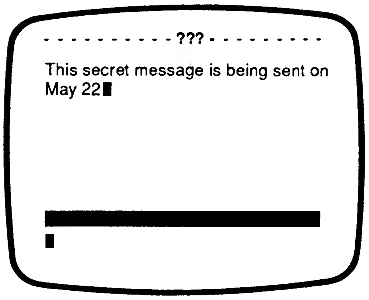
```
----------???---------
The party will be at four o'clock
```



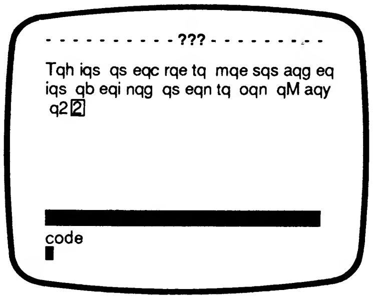
```
----------???---------
Thqe qpaqrtqy qwiqllq bqe qatq fqouqr
qo'qclqocqk

code
```

```
to code
top
repeat textlen / 2 [cf cf insert "q]
end
to decode
top
repeat textlen / 3 [cf cf delete]
end
```

- Inserting letters and spaces in the message.



```
----------???---------
This secret message is being sent on
May 22
```



```
----------???---------
Tqh iqs  qs eqc rqe tq  mqe sqs aqg eq
iqs  qb eqi nqg  qs eqn tq  oqn  qM aqy
 q22

code
```

```
to code
top
repeat textlen / 2 [cf insert "q
                    cf insert char 32]
end
```

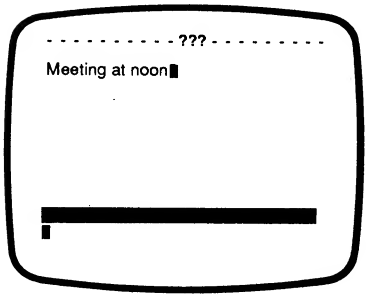
Char 32 is the space.

```
to decode
top
repeat textlen / 4 [cf delete cf delete]
end
```

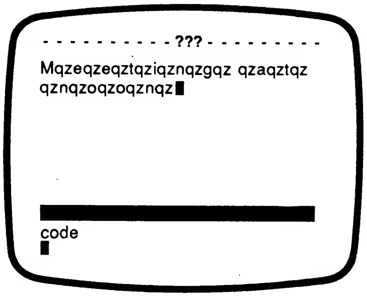
- Inserting two characters after every letter in the message.

Meeting at noon

Mqzeqzeqztqziqznqzgqz qzaqztqz qznqzoqzoqznqz



```
- - - - - - - - - - ??? - - - - - - - - -
Meeting at noon
```



```
- - - - - - - - - - ??? - - - - - - - - -
Mqzeqzeqztqziqznqzgqz qzaqztqz
qznqzoqzoqznqz

code
```

```
to code
top
repeat textlen [cf insert [qz]]
end

to decode
top
repeat textlen / 3 [cf delete delete]
end
```

Chapter **6**

Words Come Alive!

Introduction 50

PUN FUN 51

Off-computer Activity

Understanding Puns 51

Working Through the Project on the Computer

Making the Shapes 51

Positioning the Man and the Clock 52

Getting the Message Across 53

Presenting the Pun 53

Adding Sound 54

Going Further 54

ROW, ROW, ROW YOUR BOAT – A MUSIC VIDEO 55

Off-computer Activity

What is a Music Video? 55

Working Through the Project on the Computer

The Boat and the Sailor 55

Setting Things Up 55

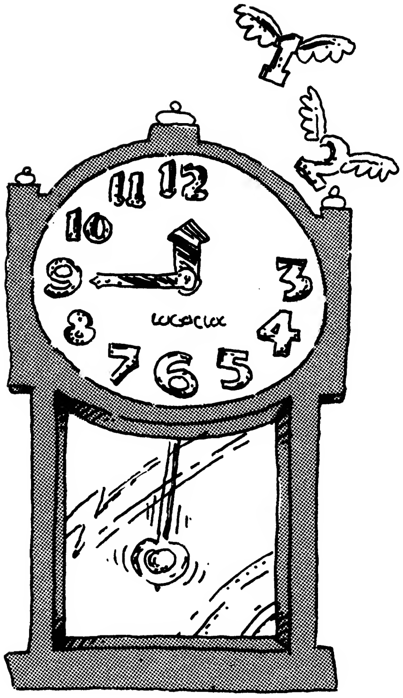
Row the Boat 56

Moving Right Along 56

Adding Music 56

Presenting the Video 56

Going Further 57



Introduction

The projects in this booklet are specifically related to language arts skills. They introduce the student to illustration of text and animation. Students can experiment with creating their own shapes for the turtle to "wear." Animation is produced by setting one or more turtles to a shape and moving it.

Pun Fun is the students' first exposure to creating shapes and using more than one turtle.

Row, Row, Row Your Boat is the first project to use a procedure with an input. Set aside some time to discuss this concept with the class before they start the project. (See *Procedures with Inputs* in Chapter 10, *Topics for Discussion.*)

The activity cards with orange tabs go with the projects in *Words Come Alive!* The following chart indicates the correspondence of cards to projects. Cards that introduce a new concept or primitive are marked with an asterisk.

Pun Fun

Tulip Garden	Creating shapes, multiple turtles.
Rocket Take Off	Creating shapes, multiple turtles.
Line Gobbler	Creating shapes.
It's Snowing!	Creating shapes, multiple turtles.
Snowman	Turtle geometry (drawing polygons), label text.
**On the Road*	Multiple turtles, creating shapes, **setpos**, **all**.
**Ferris Wheel*	Creating shapes, **setpos**, **all**.

Row, Row, Row Your Boat

One Square for All Sizes!	Procedure definition, procedures with inputs.
Gift Box	Procedure definition, procedures with inputs.
Helicopter	Procedure definition, procedures with inputs.
Traffic Signs	Turtle geometry (drawing polygons), procedure definition, procedure with an input.
A Fill-in Procedure	Procedure definition, superprocedures and subprocedures, procedure with an input.
Draw Thick Lines	Procedure definition, procedure with an input.
**Look at All Your Pages!*	Procedure definition, **pagelist**.

Pun Fun

PAGE
5

This project involves creating shapes to make a funny character and a clock that provide the illustration for a pun. The text for the pun is printed at the top of the page. Animation is introduced by having the turtle that is set to the clock shape move upwards on the page, giving the impression that "time flies."

As an extension to the project, students are shown how to use the **tone** command to make the clock "ring" when it reaches the top of the page.

Students should work through the section *How to Create Your Own Shapes* at the beginning of *Words Come Alive!* before starting the project.

Pun Fun can be integrated into a language arts activity about puns or humor in general.

Pun Fun provides experience in:

Creating shapes.
Using multiple turtles.
Defining procedures.
Superprocedures and subprocedures.

The following primitives are introduced in this project:

print
tell
tone

Off-computer Activity

Understanding Puns

Discuss the concept of a pun with the class. Make sure the students understand the definition of a pun: the humorous use of a word with two or more meanings. What kinds of puns do they like?

If you are incorporating *Pun Fun* into a classroom activity, you might want each student to write and illustrate a pun on paper first. The class can share puns and discuss ways that they could be illustrated.

Working Through the Project on the Computer

Making the Shapes

The character is composed of two shapes, one for the head and one for the body. Another shape is used for the clock. Students can design their own shapes for the character and the clock. Since this is a pun, the character should be funny.

Any students who have difficulty creating shapes can be directed to the section *How to Create Your Own Shapes* at the beginning of *Words Come Alive!* or to *The Shapes Page* Reference Card (gray section of activity cards).

- Copying a shape onto another shape number.

  If a student wants to modify a shape but still keep a copy of the original shape, the shape can be "copied" and then "pasted" onto an empty shape (shapes 1 through 10).

  1. Press the **copy** keys to keep a copy of the shape being displayed in memory.
  2. Use the **next screen** and **previous screen** keys to move to an empty shape.
  3. Press the **paste** keys to paste the shape that has been copied onto the shape grid that is displayed.

  Direct the student to *The Shapes Page* Reference Card for information on these functions.

Positioning the Man and the Clock

Turtle 0 is set to the shape for the head, and turtle 1 is set to the shape for the body of the character. One or both turtles must be moved to properly align the head and body. The **man** procedure is defined to do these steps. If necessary, you could review the steps for defining a procedure. You can also refer students to the *Defining Procedures* Reference Card.

Turtle 2 is set to the clock shape and positioned on the page.

- The turtle drawing a line when a **forward** or **back** command is given.

  To move the turtle without having it draw, use the **pu** command to lift its pen.

- Difficulty changing the turtle back to the regular turtle shape.

  The standard turtle shape is shape number 0. To change the turtle back to the turtle shape, give the command:

```
tell ⊔                  The turtle's number.
setsh 0
```

- Forgetting the number(s) of the shape(s) created for the character and clock.

  To see all the shapes, go to the Shapes page. The front side of the Shapes page contains a graphic representation of the shapes. It is automatically updated each time a shape is changed.

- Confusion between shape number and turtle number.

  You can make the following analogy: a person can wear any outfit or costume, but it is always the same person. No matter what a person looks like, we always address the person by his or her name. The turtle's number is like its name. The turtle can "wear" any shape, but you must always address the turtle by its number.

Example:

```
tell 0
st
setsh 15
fd 10
tell 1
st
setsh 11
bk 10
```

- Difficulty positioning the turtle once it has changed shape.

  When the turtle is set to any shape other than the turtle shape (shape 0), the shape does not change direction when the turtle turns. Suggest setting the turtle to shape 0 to move the turtle to the desired position on the page, and then set its shape to the new shape.

Getting the Message Across

The **print** command is used to print the text for the pun. The text is printed at the cursor position (the top of the page).

Animation is produced by moving turtle 2 (set to the clock shape) **forward 2** many times. The example given in the project is **repeat 40 [fd 2]**.

Students can experiment with varying the *speed* that the clock moves by changing the amount the turtle moves forward each time. The student can make the turtle move at a slower *speed*, for instance **repeat 80 [fd 1]**, or faster, for instance **repeat 20 [fd 4]**.

- Forgetting to enclose the input to **print** in square brackets.

  The input to **print** must be enclosed in square brackets. Otherwise, LogoWriter interprets the word following **print** as a procedure, and tries to run it. For example:

```
print Did you ever see time fly?
I don't know how to Did
```

Presenting the Pun

Make sure students understand the reason for writing the **pun** superprocedure. Without a superprocedure, they would have to type the following instructions every time they wanted to show the pun:

```
rg
ct
man
clock
pr [Did you ever see time fly?]
repeat 40 [fd 2]
```

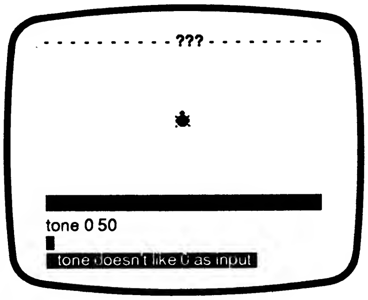
The superprocedure **pun** runs these instructions.

Adding Sound

As an extension to *Pun Fun*, the project shows how to use **tone** to add "sound effects." Encourage students to experiment with different inputs to **tone**. The first input controls the *frequency* of the pitch, and the second input controls the *time* the sound lasts. You can refer students to the *Melody* Activity Card (light green section) for more practice with **tone**.

- Using an invalid input to **tone**.

  If the input for *frequency* or *time* is above or below the limits of the computer, LogoWriter will print a message. For example:

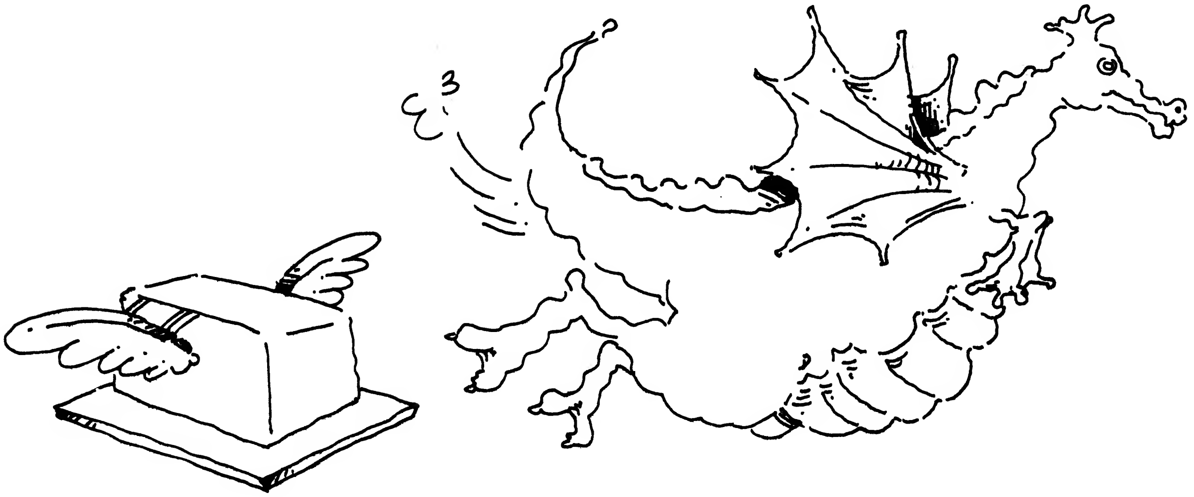


```
- - - - - - - - - - - ??? - - - - - - - - - -

tone 0 50

tone doesn't like 0 as input
```

  See the **tone** primitive in the *LogoWriter Reference Guide* for further information on the inputs to **tone**.

Going Further

- Create another animated pun. Challenge the students to think of original ideas. You can suggest the following variations on the example given in the project: Have you ever seen a butter fly? Have you ever seen a dragon fly?



Row, Row, Row Your Boat – A Music Video

PAGE
13

Row, Row, Row Your Boat uses turtle graphics commands, the **tone** command and **print** instructions to create an animated song. Two types of action are involved: the animation of the images, and the timing of the lines of music and the words of the song.

This is the first project that uses a procedure with an input. It is broader in scope than the previous projects and more open-ended. This project provides a model of how to develop a music video. Students should be encouraged to continue the song, or create another music video as a follow-up project.

Row, Row, Row Your Boat provides experience in:

Creating shapes.
Using multiple turtles.
Defining procedures.
Superprocedures and subprocedures.
Procedures with inputs.

The following primitives are introduced in this project:

seth (referred to as set heading)
wait

Off-computer Activity

What is a Music Video?

A music video involves combining music with images. Ask the students to describe music videos they are familiar with. What makes them exciting?

Timing and rhythm are important in the integration of music and images. The images should illustrate the words and music.

You can organize a class activity to clarify this process. Choose a song that all the students know. The class can sing the lines of the song, and a few students can simultaneously "act out" the words of the song. What would happen if the whole song came first, followed by the acting? Would the effect be the same?

Working Through the Project on the Computer

The Boat and the Sailor

The instructions to position the sailor in the boat will depend on the dimensions of the boat. Students should experiment with the instructions to put the sailor correctly in the boat before defining the **sailor** procedure.

Setting Things Up

To get the two halves of the boat and the sailor (turtles 0, 1, and 3) to move together, the three turtles must be heading in the same direction. Students are introduced to the **seth** (referred to as set heading) command. **Seth** sets the turtle's heading according to screen coordinates. The screen is like a compass or a map:

North
0°
West 270° —|— 90° East
180°
South

If the command **seth 0** is given, the turtle(s) will face up (north); if the command **seth 180** is given, the turtle(s) will head down (south), etc. **Seth** affects only the way the turtle is facing (heading); it does not affect the turtle's position on the screen. In the project, the following instructions set the heading of the three turtles so that they are facing the right side of the screen.

```
tell [0 3 1]
seth 90
```

Row the Boat

The "rowing" effect is produced by changing turtle 1's shape quickly between the two rowing positions. The **row** procedure is provided. Students can adjust the *speed* of the rowing by changing the input to **wait**. **Wait 20** is approximately one second.

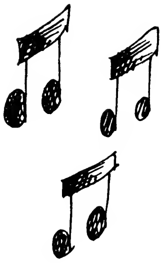
Moving Right Along

To move the boat, the two turtles set to the boat shapes (turtles 0 and 3) and the turtle set to the sailor shape (turtle 1) must all move simultaneously. The **move** procedure makes the sailor row, then moves the three turtles forward. The amount that the boat moves with each rowing movement can be changed by using different inputs to **forward**.

- The three turtles moving in different directions.

  Check the **setup** procedure to make sure that the instructions to set the heading of turtles 0, 1, and 3, to 90 are included in the procedure.

```
to setup
boat
sailor
tell [0 3 1]
seth 90 ←
end
```



Adding Music

The procedures to play the musical notes *c, d, e, f,* and *g* are provided. Each procedure uses an input for the length of time the sound lasts.

The **begin** procedure prints the first line of words and plays the first line of music. The **gently** procedure prints the second line of words and plays the second line of music.

Presenting the Video

The **video** procedure puts the music, text, and images together. The sailor rows, the first line of words appears on the page followed by the music, the sailor rows some more, the second line of words appears followed by the music, and the sailor rows some more.

Going Further

These follow-up activities are suggested in the project:

- Finish *Row, Row, Row Your Boat.*

Students will need three more musical notes to complete the song:

```
to a :time
tone 440 :time
end
```

```
to b :time
tone 493 :time
end
```

```
to c2 :time
tone 523 :time
end
```

Here are some sample procedures to finish the song:

```
to merrily
pr [Merrily, merrily, merrily, merrily ..]
c2 5
wait 1
c2 5
wait 1
g 5
wait 1
g 5
wait 1
e 5
wait 1
e 5
wait 1
c 10
wait 1
c 10
wait 1
end
```

```
to ending
pr [Life is but a dream.]
g 5
wait 1
f 5
wait 1
e 5
wait 2
d 5
wait 1
c 5
end
```

Introduction

The three projects in this booklet relate to social studies. *Walking Through Your Neighborhood* involves drawing an animated map. It reinforces an awareness of the neighborhood. *Dice 4* and *Leaping Turtle* involve creating games to be played by more than one player. They reinforce the concepts of the conventions and rules involved in game playing and also use the math concept of probability.

Walking Through Your Neighborhood introduces recursion in an extension to the project.

Dice 4 introduces **random** as a means of setting each turtle randomly to a "die side" shape – giving the same result as "rolling" a die by hand.

Leaping Turtle uses a recursive procedure to get LogoWriter to follow the rules of the game. **If** is used to create conditional instructions. This project involves more complex programming than *Walking Through Your Neighborhood* and *Dice 4*.

Introduce the concept of recursion before *Walking Through Your Neighborhood.* (See *Recursion* in Chapter 10, *Topics for Discussion.*)

The activity cards with red tabs go with the projects in *You and Your Friends*. The following chart indicates the correspondence of cards to projects. Cards that introdı ice a new concept or primitive are marked with an asterisk.

Walking Through Your Neighborhood

Climbing a Ladder	Recursion, creating shapes.
A Turtle Eraser	Programming keys (**when**).
Sound Effects - IBM Only	Procedure with an input.
Sound Effects - Apple Only	Procedure with an input.
**Table of Contents*	Word processing commands, **tab**, **printscreen**.
Caterpillar	Multiple turtles, recursion.

Dice 4

A Starry Sky	Procedure with an input.
**Quick Change*	**Color**, **shape**, **bg**.
Big Titles	Creating shapes, setting the turtle's heading.

Leaping Turtle

Spirals	Recursion, procedure with an input.
**Bring Your Page to Life!*	Defining a **startup** procedure, multiple turtles, **setpos**.
**A Rainbow!*	Multiple turtles, **setpos**, **who**.

Walking Through Your Neighborhood

PAGE 3

This project involves drawing a map that comes to life. The map consists of a street and the buildings on it. It is brought to life by having a person walk along the street. As the person passes each building, some information about the building is printed on the page.

This is a "personalized" map in the sense that students can draw any street, design the buildings any way they want, and write any text about the buildings – for instance, information about their occupants, their location, their history.

The open-endedness of this project makes it more advanced than previous projects. Not all the procedures are provided. Students must plan and develop the details of the project on their own.

As an extension to the project, students are shown how to move a truck (the turtle set to a truck shape) along the street.

Walking Through Your Neighborhood can be integrated into a social studies project on maps. The class can discuss the purposes of maps, techniques of mapping, different kinds of maps, etc. It could also be adapted to a history project – students could draw an animated map of a historically important site or area of the city or town.

Walking Through Your Neighborhood provides experience in:

Creating shapes.
Integrating text and graphics.
Superprocedures and subprocedures.
Procedures with inputs.
Recursion.

The following primitive is introduced in this project:

home

Off-computer Activity

The Components of a Map

Students may only be familiar with maps representing large areas. Introduce them to the idea of drawing a map of a small area. This kind of map could represent the street that the student lives on, the street block that the school is on, or a block on the main street of their town. Ask the students what they think could be included in this type of map. They might come up with the following ideas:

The street or streets.

The names of these streets.

The buildings on the street.

Other scenery or details of interest, such as parks, trees, monuments, even parked cars.

More information about the location of these streets; for example, the name of the town or city, perhaps even the state.

You can discuss the orientation of maps. Maps are always drawn according to the compass system, with North at the top, South at the bottom, West at the left and East at the right. You may want to suggest that students adhere to this convention when drawing their maps.

Working Through the Project on the Computer

The model given in the project booklet is designed to give students an idea of how the map works and different ways of doing each step. The project is very open-ended. The student could draw just one street with a few buildings on it or an intersection with buildings on both streets. It is probably a good idea to encourage students to start off with a fairly simple map, like the one shown in the project; a more elaborate map can be tackled as a subsequent project. Even a simple map provides students with lots of scope to individualize their projects. A student can "enhance" a simple version of a map by adding scenery and color.

Students should be able to proceed independently through the project. Remind them to define a procedure for each part of their map as they go along.

Drawing the Map

Drawing the street involves drawing the lines for the street, putting houses on it, and labeling it. These instructions should all be included in a **street** procedure.

Students can draw either one street or an intersection. Be sure that they leave enough space at the top of the page to print the text.

There are two methods for drawing the buildings. Students can use one or both.

- Turtle graphics commands.
- Stamping a shape. Students can use the house shape (shape 20) or create a new shape.

Either the **label** keys or the **label** command can be used to label the street. The **label** command is preferable, as the labeling instructions can be included in the **street** procedure.

Walking Along

The character walks along the street to each building in sequence. The **walk** procedure uses an input for **distance**. **Distance** is the number of turtle steps the character moves. When the character reaches a building, it stops and some text is printed on the page. The text should remain on the page long enough for the "observer" to read it, then be erased.

Students should define a **tour** procedure that prints, ***walks*** to the next building, prints, walks, prints, walks, etc. Here is an example of a **tour** procedure:

```
to tour
pr [This is my house.]              LogoWriter prints information
pr [My address is                   on the page.
   5855 McLynn Avenue.]
wait 200                            LogoWriter waits.
ct                                  LogoWriter clears the text.
walk 35                             The person walks.
pr [This is Nicole's house.]        LogoWriter prints.
pr [She is my best friend.]
wait 200
ct
walk 45                             The person walks.
pr [This is the library.]           LogoWriter prints.
wait 200
ct
end
```

If the student has a lot of buildings on the street, suggest breaking each "segment" of the **tour** procedure into a subprocedure. For instance:

```
to tour
myhouse
walk 35
nicolehouse
walk 45
library
end

    to myhouse
    pr [This is my house.]
    pr [My address is 5855 McLynn Avenue.]
    wait 200
    ct
    end

    to nicolehouse
    .
    .
    .
    end

    to library
    .
    .
```

Adding a Moving Truck

As an extension to the project, students can set a turtle to a car or truck shape and move it along the street. The project shows how to define a simple recursive procedure to move the truck:

```
to move
fd 1
move
end
```

Tell the students to press the **stop** keys to stop the procedure.

Students may suggest another way to move the truck – giving the instruction to repeat a forward movement many times, as was used in *Pun Fun* to move the clock. For example, **repeat 50 [fd 2]**. Both methods of moving the turtle are acceptable.

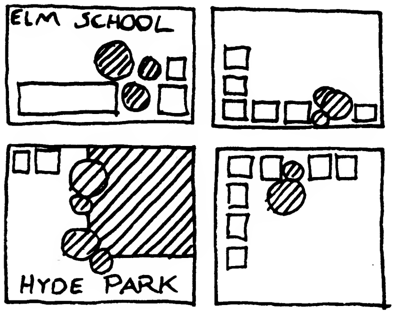
Wrap-up Activity

When the students have completed their maps, you may want to discuss how a map drawn on a computer is different than a map drawn on paper. They might think of the following points:

- A shape can be created to represent something such as a building, a park, a tree, or a vehicle, and then stamped on the page. The same shape can be stamped a number of times.
- A procedure can be defined for each part of the map. This makes it easy to change any part of the map.
- Only a computer version of a map can "come to life."

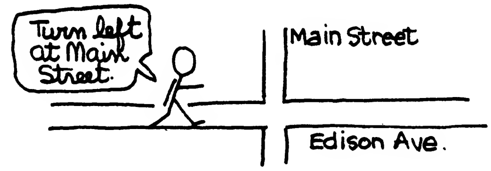
Going Further

- Students can enhance their maps by adding scenery and color. Examples: stamped trees, green squares or rectangles for parks, a sun

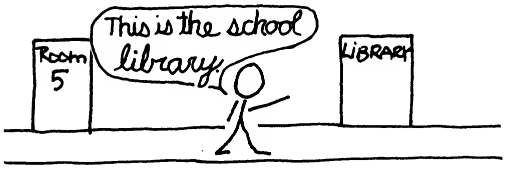


- Students can draw another animated map. Here are some ideas:

A map showing how to get to a particular place. The "person" could walk the route, giving information about the route or about buildings passed on the route.



A map of the interior of the school. The "person" could walk down the corridor giving information about each room passed.



A map of the buildings in a historic site. Information about the history of each building could be printed on the page. This situation is almost like a "walking tour" with a guide giving information about each building.

- As a class project, students could combine their efforts to draw a large map of the neighborhood. Each student could do a "segment" of the map. This project would require a lot of advance planning:

  Drawing a sketch of what will be included in the map.

  Dividing the work among the students.

  Deciding on the size of each part of the map. For example, the road must always be the same width so that the parts can be put together.

When all the parts of the map are complete, students can print them out and put the map together.

Dice 4

PAGE 13

This project involves setting up a dice game, called *Dice 4*, on the computer. Shapes 1 through 6 are made to represent the six sides of a die. Each of the four turtles is a die. Setting each turtle's shape randomly to a shape between 1 and 6 gives the same result as "rolling" a die by hand.

A lot of the fun of this project is in the result – having a game on the computer to share with friends.

Dice 4 can be integrated into a social studies project on games. An important element of game playing is following the rules. Who decides the rules of a game? Can rules be changed? How do rules affect game playing strategies? Students may also be interested in the history of games and the role of games in society.

This project involves the concept of probability. Probability is introduced through the use of random numbers.

Dice 4 provides experience in:

Creating shapes.
Superprocedures and subprocedures.
Procedures with inputs.

The following primitives are introduced in *Dice 4:*

all
each
random

Off-computer Activities

Randomness and Probability

Discuss the probability of events that students can relate to. Examples:

- What is the probability that a rolled die will come up 6 or 1 or 2?
- What is the probability that four dice will all come up 6?
- What is the probability that a card pulled out of a full deck of cards will be a Queen?
- What is the probability that a flipped coin will come up heads?

You can reinforce the concepts of randomness and probability by rolling a real die. Let the students take turns rolling the die, and record the number that comes up each time. If the die is rolled 100 times, how many times does it come up 1, 2, 3, 4, 5, 6? Later, when the students have set up the turtle dice on the computer, they can roll a die on the computer 100 times. A comparison can be made between the number of times each number came up on the computer die and on the real die.

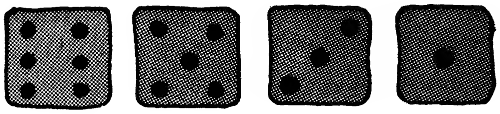
How to Play Dice 4

It is important that the students understand how to play *Dice 4* before they set up their own game on the computer. Probably the easiest way to explain the game is to show the students how to play it using four dice.

Demonstrate the game with two players. The object of the game is to score the highest number of points. Each player gets three rolls. After his or her three rolls, the player adds up the dice.

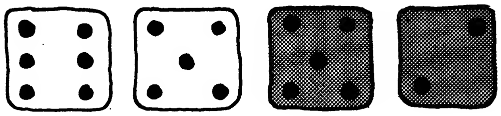
The first player rolls the dice, then looks at the four dice. The player keeps the die or dice with the highest numbers, and rolls the remaining die or dice. For example:

First turn:



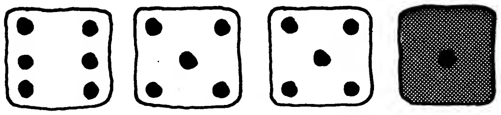
This player has a 6, a 5, a 3 and a 1. The best strategy will be to leave the 5 and the 6 in place and roll the other two dice.

Second turn:



The player decides to roll only the die with a 2.

Third turn:

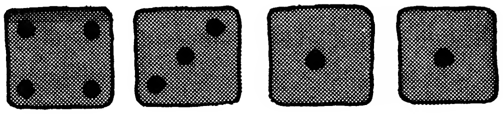


6 + 5 + 5 + 1 = 17

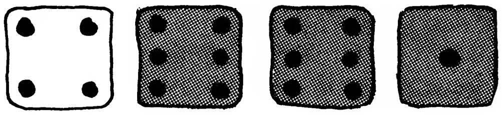
This player scored 17.

When the first player has completed three rolls, it is the second player's turn. For example:

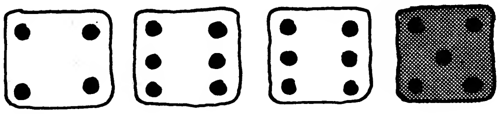
First turn:



Second turn:



Third turn:



4 + 6 + 6 + 5 = 21

Player 2 is the winner!

Planning a Computer Version of Dice 4

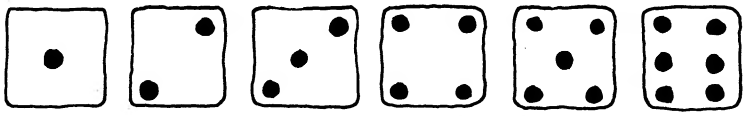
Explain to the students that the only thing that is different about playing *Dice 4* on the computer is that LogoWriter "rolls the dice" for them. Each turtle will be a die. The player simply tells LogoWriter which die or dice to roll.

On the first roll of each turn, the player will tell LogoWriter to roll all four dice. On the second and third rolls, the player must decide which die or dice he or she wants to roll, and then give LogoWriter this instruction. LogoWriter doesn't understand the rules of the game; all it can do is roll the dice you ask it to.

Working Through the Project on the Computer

Making the Shapes

Students must use shapes 1 through 6 to represent their corresponding dice numbers; otherwise the game won't work.



shape 1 shape 2 shape 3 shape 4 shape 5 shape 6

If the student wants to keep any of the existing shapes 1 through 6 before making the dice shapes, the original shapes can be "copied" onto other shapes. Refer students to the explanation *How to Copy or Erase Shapes* at the beginning of *Dice 4*.

Rolling One Die

To roll a die, the turtle is set to one of the die-side shapes (shapes 1 through 6). **Random 6** randomly selects a number less than 6 (0 through 5) and reports this number to **setsh**. Zero is included in the numbers **random** may report. Since 0 isn't a die shape, 1 must be added to the number that **random 6** reports. The instruction **setsh 1 + random 6** sets the shape to a shape number 1 through 6.

All Four Dice

Each die must be rolled independently. The **each** command is used to give each turtle a turn to roll.

The instruction:

```
each [roll]
```

tells LogoWriter *to roll each die separately*. The instructions:

```
tell all
each [roll]
```

are equivalent to:

```
tell 0
roll
tell 1
roll
tell 2
roll
tell 3
roll
```

Similarly the instructions:

```
tell [0 1]
each [roll]
```

are equivalent to:

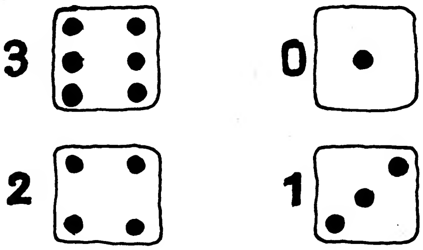
```
tell 0
roll
tell 1
roll
```

If **each** is omitted, as in:

```
tell all
roll
```

All the turtles are set to the same shape number.

It is important to label the number of each turtle, to avoid confusion about which turtle is which.



Getting Ready to Play

The **play** procedure takes, as input, the numbers of the dice (turtles) to be rolled, and rolls *each* of these dice separately.

Playing the Game

The game is ideally played by two players. Students can play *Dice 4* any number of times, adding up each player's scores to determine the winner.

- Rolling the wrong die or dice.

  If a student accidently rolls the wrong die or dice, he or she can just set the die or dice back to its original number (shape) and then take his or her turn again. For example:

```
play 1                      Suppose this is an error.
tell 1
setsh ⊔                     Set it back to the shape representing
                            its original number.
play [2 3]                  Make the correct roll.
```

Going Further

These follow-up ideas are suggested in the project:

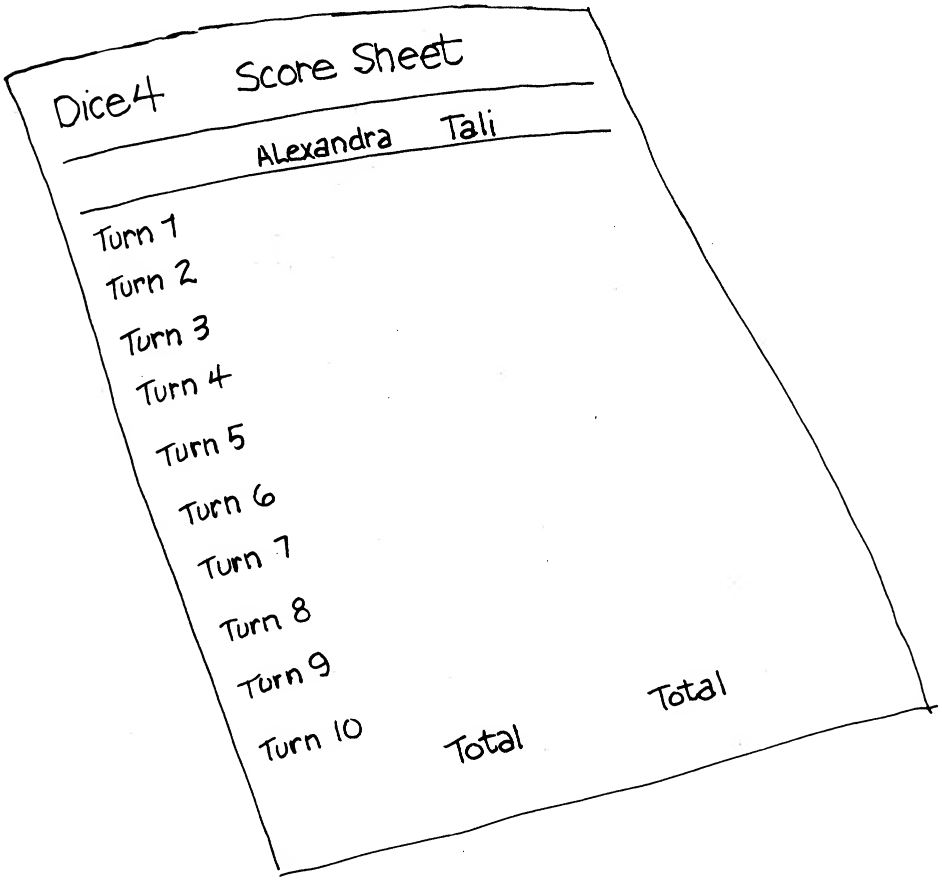
- Make up different rules for the game:

  The player with the lowest score wins.

  The player who scores closest to 13 wins.

  Play 10 times and add each person's total scores. The person who has the highest total wins.

- Make a score sheet with LogoWriter.



Leaping Turtle

PAGE
20

As mentioned in the introduction to this chapter, *Leaping Turtle* involves more complex programming than the other two projects in the booklet. **Look over the project carefully before you introduce it to the students.**

Leaping Turtle involves setting up a simple board game. The board is composed of shapes stamped in different colors. **Random** is used to move the turtle "by chance," as if the player was throwing a die to determine how many places the turtle would move. The rules of the game are determined by the color of the marker that the turtle lands on. **Colorunder** tests the color of the marker under the turtle. **If** is used to create conditional instructions in a recursive procedure – it tells LogoWriter "***if*** the turtle lands on a marker stamped in a particular color, do such and such."

As with *Dice 4*, a lot of the fun of the project is in the result – having a game on the computer to share with classmates.

Leaping Turtle provides experience in:

Creating shapes.
Superprocedures and subprocedures.
Recursion.
Using conditional instructions.

The following primitives are introduced in this project:

colorunder
if
setpos
stop

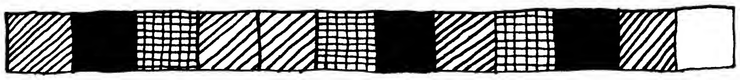
Off-computer Activity

Formalizing the Rules of the Game

The most difficult aspect of *Leaping Turtle* is telling LogoWriter how to follow the rules. To help students understand how the rules are formalized for LogoWriter, illustrate how rules that they are familiar with might be worded. Start by discussing "rules" in the child's life. Give the students one example, and see if they can think of others. Make sure they start their rules with an *if* statement. Examples:

If your team wins	[everyone will get a ribbon]
If you finish your homework	[you can watch TV]

Draw the markers for a board game on the blackboard. Fill the squares in three colors, but leave the last one empty.



Note: The patterns represent different colors.

Ask the students to formalize the rules of the game. Here are some possibilities:

- If you land on the red square, go forward three markers.
- If you land on the blue square, go back two markers.
- If you land on the white square, stay where you are.
- If you land on the empty square, you win.

Now help the students set up these rules in LogoWriter. Give one example:

if color = red [do something]

It is up to you to decide what the turtle should do. In this case, it would probably be to go forward some amount.

Tell the students:

- LogoWriter can use **colorunder** to check the color under the turtle.
- The action following the **if** statement must be enclosed in brackets.

The LogoWriter instruction would look like this.

if colorunder = ⌴ [fd 3]

↑

The color number that represents red.
(Remind the students that LogoWriter only knows colors by number.)

Now let them set up the remaining rules. This exercise is very useful for understanding the procedures in the computer version of the game. For example:

if colorunder = ⌴ [bk 2]

if colorunder = ⌴ [pr [You win!]]

if colorunder = ⌴ [fd 0]

↑

This means stay where you are.

Working Through the Project on the Computer

Note: **Be sure to tell students that in the illustration of the game board in the project booklet, different patterns are used to represent the different colors.**

It is a good idea for students to experiment with **colorunder** and = before they start drawing the game board.

Students can experiment with **colorunder** by placing the turtle on a stamped shape or filled area, and then typing:

```
show colorunder
```

Colorunder reports the color under the turtle. This color number is printed in the command center.

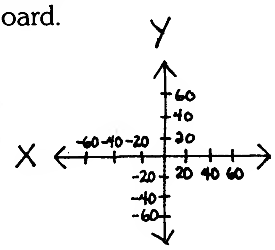
The = sign means "equals," just as it does in arithmetic. To see how it works, students can try examples such as the following:

```
show 5 = 6
show 5 + 5 = 3 + 7
show 3 * 3 = 10
```

Drawing the Board

The markers are drawn by stamping the circle shape in different colors. The winner's marker is a hollow square.

Setpos is used to position the turtle at the left side of the page, so that the markers can extend to the right across the page. **Setpos** sets the turtle according to a position in terms of cartesian coordinates. You may want to illustrate this on the board.

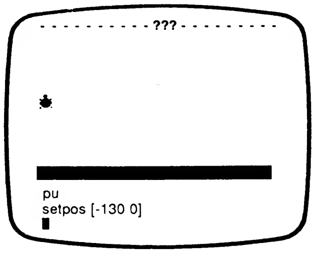


It is important that all the markers are stamped 20 turtle steps apart. Students should draw the markers as illustrated in the project.

- Confusion over the instructions to set the turtle's position and its heading.

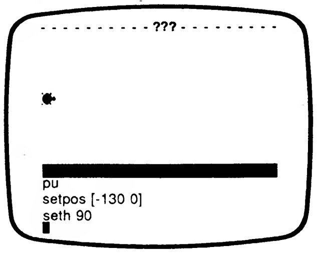
  Setpos sets the turtle's position on the page. It does not affect its heading. Likewise, the **seth** command sets the turtle's heading (the direction it is facing) but does not affect its position.

  Setpos [−130 0] positions the turtle at the left side of the page.



```
pu
setpos [-130 0]
```

Seth 90 sets the turtle's heading (the direction it is facing) to 90.



```
pu
setpos [-130 0]
seth 90
```

Moving

The **start** procedure places the turtle at the starting marker. The **move** procedure uses **random** to move the turtle forward a random number of markers, as if "by chance."

```
repeat random 3 [fd 20]
```

Reports a number less than 3 (0 through 2) to **repeat**.	There are 20 turtle steps between markers.	The turtle will move 0, 1, or 2 markers.

Checking the Color

Students may have difficulty understanding the **check** procedure. Explain that the purpose of the **check** procedure is to get LogoWriter to follow the rules of the game. LogoWriter must know:

- When the turtle lands on a marker stamped in color 2, it must jump back two markers.
- When the turtle lands on a marker stamped in color 3, the turtle jumps ahead three markers.
- If the turtle lands on color 0 (the hollow marker), the message "You Win!!" is printed on the page.

Check is recursive. It keeps calling itself as a subprocedure until it is stopped by a stop rule. A series of **if** statements are used along with **colorunder** to test the color under the turtle. LogoWriter runs one line of the procedure, then goes on to the next line of the procedure, runs it, and so on.

```
if colorunder = 0 [pr [You Win!!] stop]
```

If the color under the turtle is 0.

Print You Win!!
Stop the procedure.

If the color under the turtle is not 0, LogoWriter goes on to the next line of the procedure.

```
if colorunder = 1 [stop]
```

If the color under the turtle is 1.

Stop the procedure.

If the color under the turtle is not 1, LogoWriter goes on to the next line of the procedure.

```
if colorunder = 2 [jumpback]
```

If the color under the turtle is 2.

Run the **jumpback** procedure and then go on to the next line of the procedure.

If the color under the turtle is not 2, LogoWriter goes on to the next line of the procedure.

```
if colorunder = 3 [jumpahead]
```

If the color under the turtle is 3.

Run the **jumpahead** procedure and then go on to the next line of the procedure.

If the color under the turtle is not 3, LogoWriter goes on to the next line of the procedure.

```
check
```

Check runs itself as a subprocedure.

Check has two "stop rules." It stops if the color under the turtle is 0 or 1. You might want to show students how the procedure works:

If the turtle lands on a marker stamped in color 2, LogoWriter goes through every line of the procedure until it gets to the line:

```
if colorunder = 2 [jumpback]
```

LogoWriter runs the **jumpback** procedure. The turtle jumps back two markers and lands on a marker stamped in color 3. LogoWriter goes on to the next line of the procedure:

```
if colorunder = 3 [jumpahead]
```

LogoWriter runs the **jumpahead** procedure, and the turtle jumps ahead three markers. It lands on a marker stamped in color 1. **Check** runs itself as a subprocedure. LogoWriter goes through each line of **check** until it gets to the line:

```
if colorunder = 1 [stop]
```

The procedure stops.

Playing the Game

The game is ideally played by two players. The players can take turns running the **play** procedure.

Going Further

- If the computer has more than three colors, students can add different colored markers and change the rules in the **check** procedure.
- Students can use LogoWriter to make a score sheet to keep track of the players' scores.

Chapter **8**

News, Views, and Information

Introduction 76

LIGHTS, CAMERA, ACTION 77
Off-computer Activity
Commercials 77

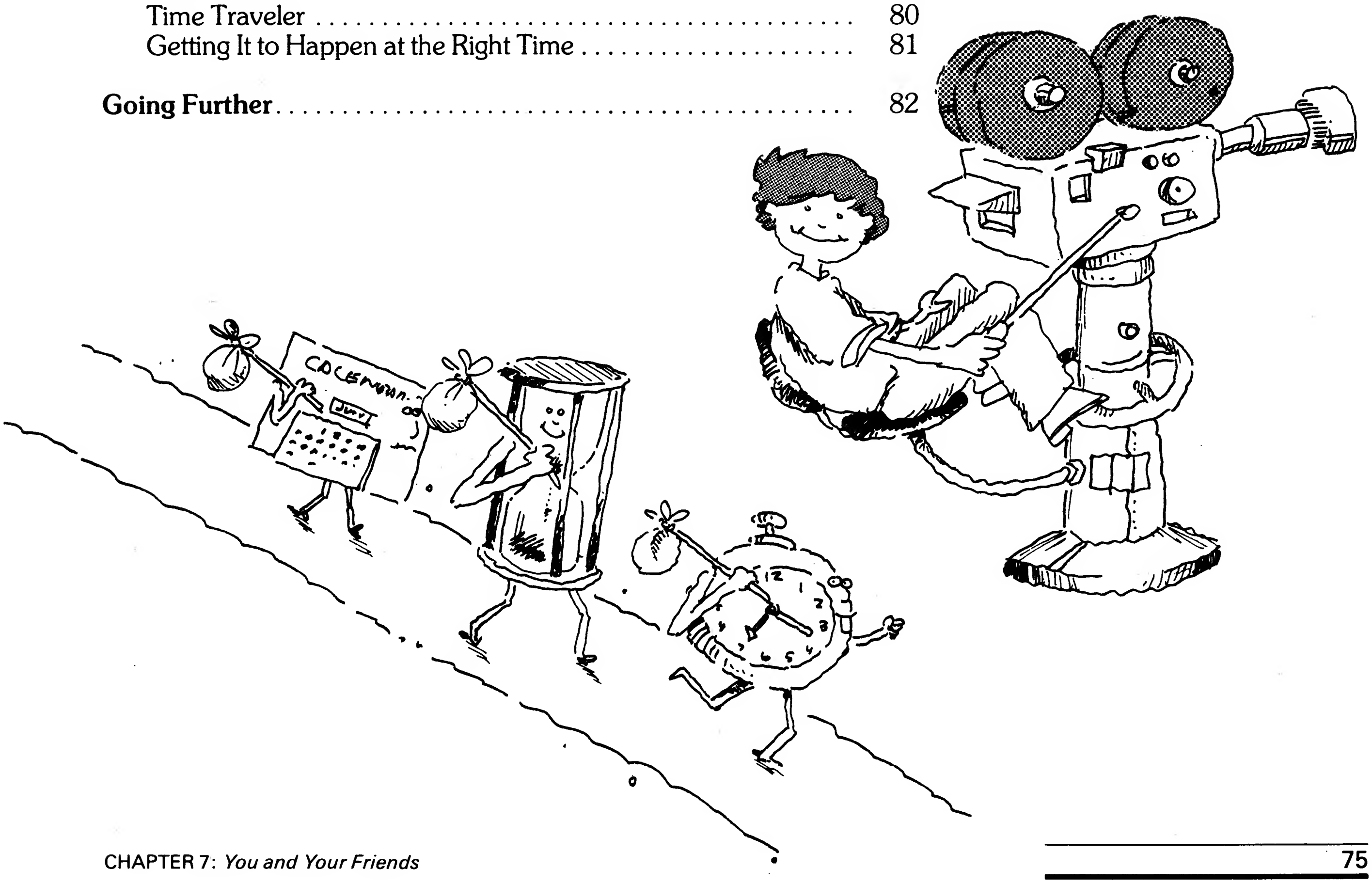
Working Through the Project on the Computer
Completing the Scenes. 78
Starting Automatically. 78

Going Further. 78

TRAVELING THROUGH TIME 79
Off-computer Activities
Introducing Time Lines. 79
Planning the Computer Version of *Time Line* 80

Working Through the Project on the Computer
Event Markers and Dates 80
The Events. 80
Time Traveler 80
Getting It to Happen at the Right Time 81

Going Further. 82



Introduction

The two projects in this booklet relate to communicating information. Both projects require a certain amount of research and advance planning and emphasize sequencing of events.

Lights, Camera, Action involves creating a commercial. A few pages are presented, one after the other, giving the impression of a sequence of events. A **startup** procedure defined on each page automatically starts the action when a page is shown.

Traveling Through Time involves presenting information about a series of important events, plotted on a line representing a period of time. Developing the time line involves the use of turtle graphics and word processing. The programming technique of conditional instructions in a recursive procedure is applied.

Lights, Camera, Action and *Traveling Through Time* are more open-ended than previous projects. They provide students with the opportunity to apply skills and concepts learned in previous projects.

The activity cards with purple tabs go with the projects in *News, Views, and Information.* The following chart indicates the correspondence of cards to projects. Cards that introduce a new concept or primitive are marked with an asterisk.

Lights, Camera, Action

A Talking Clown	Creating shapes, multiple turtles, recursion.
Address Book	Word processing.
UFO!	Multiple turtles, creating shapes, setting the turtle's heading and position.
Shark!	Creating shapes, turtle graphics.

Traveling Through Time

Stop Rules: Spirals	Stop rules in a recursive procedure, procedure with an input, arithmetic reporters.
The Countdown	Stop rules in a recursive procedure, procedure with an input.
Turtle Tag	Programming keys, sound.
Turtle Squash	Programming keys, conditional instructions.
**Startup Your Scrapbook*	Naming a page startup.
**Change Your Mind?*	Text editing commands, recursive procedure with two inputs, **found?**

Lights, Camera, Action

Making a commercial involves creating a series of "frames" that can be shown in sequence, giving the impression of a sequence of events. In this LogoWriter version of a commercial, each frame is a page. LogoWriter gets a page, some action occurs, LogoWriter gets another page, some action occurs on that page, etc.

Lights, Camera, Action provides students with lots of scope. They should be encouraged to add other features to their commercials besides the ones given in the project. A student may take a number of sessions to complete the project.

This project can be integrated into a language or communication arts activity on commercials and film.

Lights, Camera, Action provides experience in:

Integrating text and graphics.
Defining a procedure with two inputs.
Defining a **startup** procedure.
Getting pages under procedure control.

Off-computer Activity

Commercials

Start off by discussing what a commercial is: an advertising broadcast on radio or television. If you are incorporating *Lights, Camera, Action* into a class discussion or project on commercials, you may want to discuss these ideas:

The history of commercials.

The techniques for producing commercials and film.

How commercials influence the consumer.

How the "target audience" of a commercial is defined.

The positive and negative effects of commercials on different groups, for instance, children.

The techniques used in commercials for communicating ideas in a short time frame.

Discuss the "essential ingredients" of a commercial:

It must try to convince people to do something.

It must have a short and cohesive script.

The sequence of events should be logical.

It must transmit a clear message.

Working Through the Project on the Computer

Completing the Scenes

Students should complete the two scenes shown in the project, and then either add other scenes to complete this commercial or create another commercial.

For each of the pages of the commercial developed in the project, the student is shown how to define a **background** procedure to draw the background and an **action** procedure to carry out the action (the animation on the page). This organization helps structure the elements of the project.

Using the same procedure name on different pages to do different things is an important feature of LogoWriter. Primitives can be used on any page; they are built into LogoWriter. Procedures are local to a page; a procedure can only be run if it appears on the flip side of the page that is showing. For this reason, the same procedure name can be used on different pages.

For subsequent pages of the commercial, students can continue this organization of having a procedure named **background** to set up the background scene for the particular "frame," and a procedure named **action** to perform the action.

Students should not find the programming level of the procedures in this project too difficult. They will probably encounter small bugs, such as placing the turtle incorrectly, moving the turtle too quickly, forgetting to put the pen up or down at the right times, confusing which turtles are being addressed, etc. Encourage them to "debug" these errors independently.

Starting Automatically

The commercial is run by "getting" each page of the commercial in sequence. A **startup** procedure should be defined on the flip side of each page. The **startup** procedure runs the action. When LogoWriter gets each page, it is shown with the background scene. The action starts automatically.

Refer students to the *Bring Your Page to Life!* activity card (red section) for more practice using a **startup** procedure.

Going Further

- Add other scenes to the commercial.
- Create a different kind of commercial. Some ideas are given in the project.

I USED TO BE BALD
UNTIL I DISCOVERED GLOW-N-GROW SHAMPOO!
SHAMPOO
NOW I HAVE A FULL HEAD OF CURLS

- A group of students can combine efforts to make a longer commercial. In this case, each student can design a "page" for the commercial.

Traveling Through Time

Traveling Through Time develops a dynamic time line of early American history. The emphasis of the project is on the relationship of time to events. *Traveling Through Time* reinforces the concept of time measured by space, and as well incorporates a number of language arts activities. Students can write text (compositions or poems) about events on their time lines. As a subsequent project, students can develop a time line for a particular period, country, or important person being studied in history class.

The time line presented in the project booklet involves placing "letter markers" at intervals on the line. The markers represent important years in the early history of the United States.

A "time traveler" (the turtle) moves along the time line. When the turtle gets to a letter marker, it stops, and something happens. The letter under the turtle indicates to LogoWriter what should happen. For instance, LogoWriter might get another page, where something is written about the event that happened on the date represented by the marker.

Traveling Through Time provides experience in:

Integrating text and graphics.
Getting pages under procedure control.
Using conditional instructions.
Recursion.
Using a **startup** procedure.

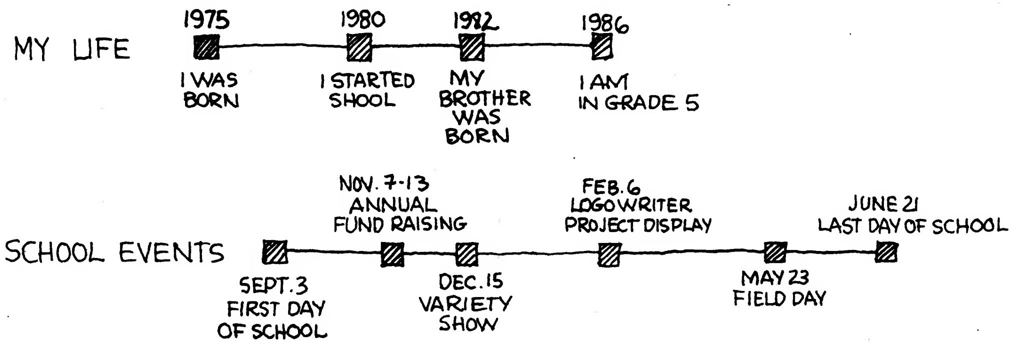
The following primitives are introduced in this project:

cb
charunder
cursorpos
search

Off-computer Activities

Introducing Time Lines

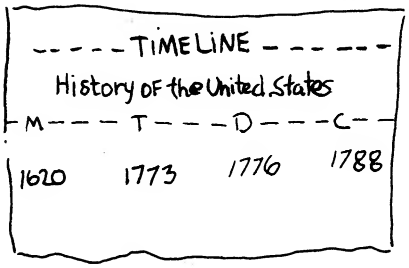
If the students have never done a time line before, be sure to explain the function of a time line. A time line is simply a presentation of important events in sequence. The events can be any kind of data. Examples: events in the student's own life, events in history, events throughout the school year, important holidays during the year. You could draw sample time lines on the blackboard for the students.

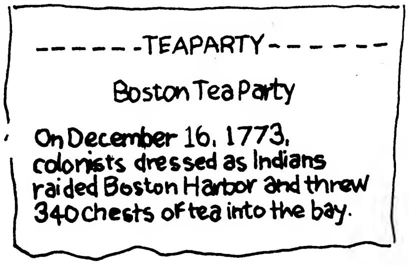


You should look carefully at the time line in the project booklet before introducing this project to the students. While the programming concepts introduced in the project are not too difficult, the steps involved to complete the time line are more complex than the steps involved in previous projects.

Planning the Computer Version of Time Line

Introduce the students to the idea of a dynamic time line as presented in the project booklet. This time line is a line with letter "markers" on it.





On December 16, 1773, colonists dressed as Indians raided Boston Harbor and threw 340 chests of tea into the bay.

The turtle moves along the time line. Each time it crosses a marker, the turtle stops and LogoWriter runs a procedure describing the event. The letter under the turtle indicates to LogoWriter which procedure to run.

Give students examples of ways the events could be described: stories, poems, pictures.

Working Through the Project on the Computer

Event Markers and Dates

Letters are used as markers on the time line. Remind students that they should choose letters that represent the events in some way. For example, M for Mayflower. Each letter can only be used once. Hyphens are used to space the letters evenly and give the visual impression of a line. LogoWriter adds a space before and after every hyphen.

To put in the years, students can use the **print** command as in the project booklet, or the **label** command.

The Events

Each event must be described or illustrated on a separate page. A procedure must be defined for each event. Each "event" procedure:

- Gets the page with the description of the event.
- Returns to the timeline page.

Two examples of procedures for events are provided. **Remember: The procedures describing the events must be defined on the timeline page.**

Time Traveler

The **start** procedure places the turtle at the beginning of the line. Word processing commands are used to place the cursor on the first hyphen on the page (the beginning of the line). The instruction **setpos cursorpos** sets the turtle's position to the cursor position.

Getting It to Happen at the Right Time

The **travel** procedure uses a series of **if** statements along with **charunder** to test the character (letter) under the turtle. When the turtle crosses a letter marker, something happens. *If* the character under the turtle is m, LogoWriter runs the **mayflower** procedure, *if* the character under the turtle is t, LogoWriter runs the **teaparty** procedure, etc. See the explanation of the **check** procedure in *Leaping Turtle* for more information on this programming technique (use of conditional statements in a recursive procedure).

- The turtle moving upwards on the page, rather than along the time line.

  Check that the instruction **seth 90** is included in the **start** procedure.

  ```
  to start
  top
  search "-
  setsh ⎵
  pu
  setpos cursorpos
  seth 90 ←
  end
  ```

- The **travel** procedure doesn't stop.

  Check the **travel** procedure to make sure the **stop** command is inside the instructions for the last event. For example,

  ```
  if charunder = "c [constitution fd 10 stop]
  ```

- Confusion about why the instruction **fd 10** is given after each event in the travel procedure.

  For example:

  ```
  to travel
  if charunder = "m [mayflower fd 10]
  if charunder = "t [teaparty fd 10]
  .
  .
  .
  .
  ```

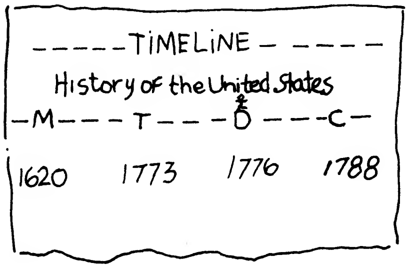
  The turtle must be moved off the marker after the procedure describing the event for that marker is run; otherwise LogoWriter will run the procedure describing the same event again and again. The width of one character is approximately 10 turtle steps. In the above example, if the turtle is over the m, LogoWriter runs the **mayflower** procedure. **Travel** is recursive. If the instruction **fd 10** is not included, the turtle will not move off the m marker. Each time LogoWriter gets to the line

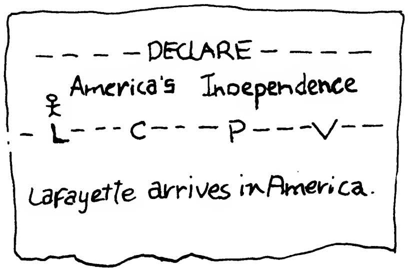
  ```
  if charunder "m [mayflower]
  ```

  it will run the **mayflower** procedure again.

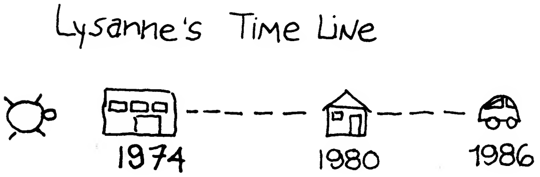
Going Further

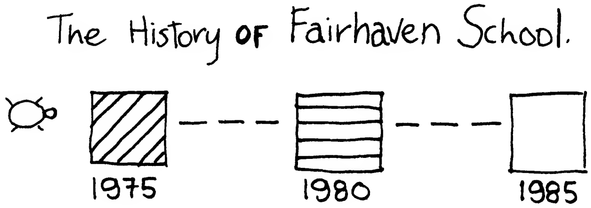
- A time line can have many layers. For instance, when the declaration is signed, the turtle can branch off to a time line of events about the Declaration and the Revolutionary War. When the constitution is ratified, the turtle can branch off to a description of the constitution. This kind of extension is ideal for a group activity.





- See if students can think of different ways to draw time lines using LogoWriter. Example: Use stamped shapes in different colors as markers. **Colorunder** can be used to test the color under the turtle.





Chapter **9**

Get the Facts!

Introduction 84

CONVERSATIONS 85

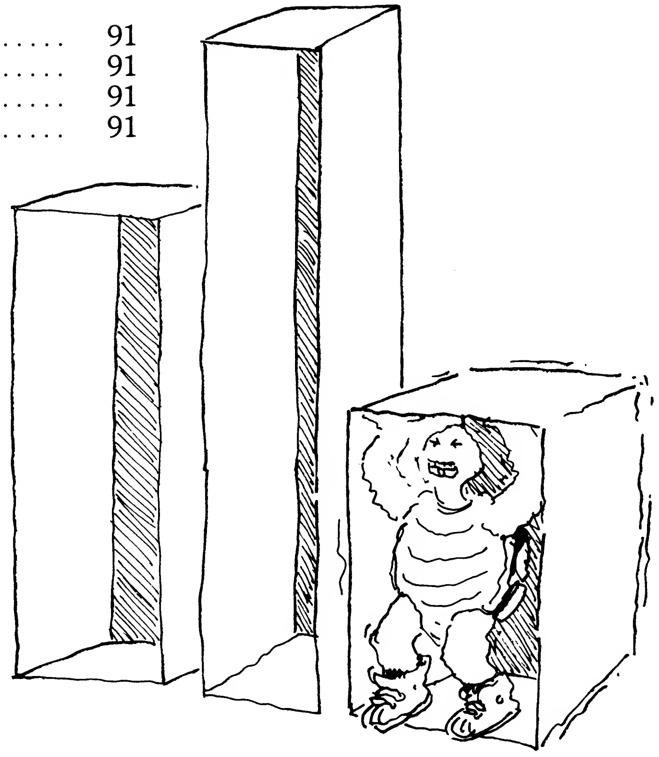
Going Further 85

TAKING A SURVEY 85
Off-computer Activities
What is a Survey? 86
A Blackboard Version of a Survey 86
Planning the Computer Version of Survey 87

Working Through the Project on the Computer
Questions and Answers 88
Keeping a Permanent Record 88
Starting Automatically. 89
The Results 89

BAR GRAPHS 90
Off-computer Activities
What is a Graph?. 90
A Blackboard Version of a Graph 90

Working Through the Project on the Computer. 91
Drawing a Graph. 91
Labeling the Graph. 91
The Bars. 91



Introduction

The projects in this booklet show the student how to create an interactive program, set up a survey on the computer, and draw a bar graph. Students should work through *Conversations* before starting on *Taking a Survey*.

Conversations is the first project to use variables. Introduce variables before the students start *Conversations*. (See *Variables* in Chapter 10, *Topics for Discussion.*)

The activity cards with the blue tabs go with the projects in *Get the Facts!* The following chart indicates the correspondence of cards to projects. Cards that introduce a new concept or primitive are marked with an asterisk.

Taking a Survey

**Turtle Doodling*	Moving the turtle around the page using four keys. **Readchar**.
**Hopping Turtle*	Using keys to control the turtle. Procedures with inputs, recursion and stop rules, **readchar**.
**Guess a Number*	A number guessing game. Conditional statements in a recursive procedure, **readchar**.

Bar Graphs

Bar Charts	A bar chart using rectangles for the bars. Procedure with two inputs.
Cartesian Grid	A grid for plotting graphs.

The following cards use fairly complex programming techniques. Make them available to students who are ready for more advanced programming.

3-D	Using trapezoids to create the illusion of depth. Recursion and a stop rule.
Spiral In	Drawing spirals with four turtles. Procedure with two inputs, recursion.
**Beyond Polygons*	Fancy polygons in multiple colors. Recursive procedure with three inputs and a stop rule.
**Digital Counter*	A Digital Counter using turtle shapes for numbers. Recursion, **ifelse**.
**Bumper Turtles - Apple Only*	Use of game paddles to play a bumper game with two turtles. Recursion, **paddle**.

Conversations

PAGE 3

Conversations shows how LogoWriter can print a question on the page, and "read" or receive the answer typed in at the keyboard. LogoWriter then uses this answer in a sentence. This kind of interaction between a person and LogoWriter is often referred to as "interactive programming." Interactive programming is used to create programs students may have used in school and at home, such as games and quizzes.

Conversations is tutorial in style. Students should be able to work their way through it fairly independently.

Conversations provides experience in:

Defining an interactive procedure.
Using names (variables).

The following primitives are introduced in this project:

name
readlist
sentence

Going Further

Encourage students to write a longer "conversation" procedure. For instance:

```
to inquire
pr [What is your name?]
pr [Press Return when you finish typing.]
name readlist "person
pr [Where do you live?]
pr [Press Return when you finish typing.]
name readlist "place
ct
pr sentence [Hi there] :person
pr sentence [I hope you like] :place
end
```

Taking a Survey

PAGE 7

This project involves planning a survey and collecting the data. LogoWriter asks the questions and stores the answers. The student accesses this data and computes the averages. LogoWriter's arithmetic reporters can be used to do the calculations.

Taking a Survey provides experience in:

Defining an interactive procedure.
Using names (variables).
Using a **startup** procedure.
LogoWriter arithmetic reporters.

The following primitives are introduced in this project:

newpage
tab

Off-computer Activities

What is a Survey?

It is important that the students understand what a survey is: a sampling of facts, figures, or opinions. Surveys are usually used to determine trends. Most students will be familiar with surveys in newspapers. See if they can think of different subjects for surveys. Examples:

Who is most likely to win an election?

What television shows will stay on the air?

How people feel about a political or social issue. For instance: Should a forest be turned into a parking lot?

Consumer habits. For instance: How many families buy liquid laundry detergent, how many buy powdered laundry detergent?

How are surveys conducted? Students will probably be familiar with paper-and-pencil surveys conducted by an interviewer in a store or surveys done over the phone. Can they think of other methods of conducting a survey?

You might also want to discuss how surveys affect people. Surveys can change people's voting, shopping, viewing, and thinking patterns.

Students can look in the newspaper for results of surveys.

A Blackboard Version of a Survey

If your class has experience collecting information and calculating averages, you can proceed directly to the next section, *Planning the Computer Version of Survey*. For students who do not have this kind of experience, probably the easiest and clearest way to introduce this project is to start off with a blackboard version of a survey.

Ask the students for ideas of what information would be interesting to collect in a survey. Then choose a subject to use for your blackboard version. Choose an example that is similar to the example given in the project booklet, such as the number of hours boys and girls spend doing homework each month. Ask the students to outline the steps involved in conducting this survey.

1. Decide on the question(s).

How many hours do you spend doing homework each month?
Are you a boy or a girl?

2. Collect the data.

Ask each student in the class to answer these questions. Use the following system for marking down the answers on the board.

Hours	Identity
16	Girl
14	Girl
13	Girl
17	Boy
8	Boy
16	Girl

... and so on.

3. Calculate the results.

Explain to the students that the results of surveys are often presented in terms of averages. Averages are used to avoid having to present large numbers. If the students don't know how to calculate averages, you might want to take some time to explain the concept and method involved. Completing the project will clarify the process.

- Calculate the average number of hours the students (boys and girls together) spend on homework per month. You could ask a student to do the calculations on the computer:

```
show 16 + 14 + 13 + 17 + 8 + 16 + 21 + 16 + 18 +
. . . . . . . . . .
```

```
show total number of hours / number of students
16
```

As a group, the students spend an average of 16 hours a month doing homework.

- Calculate the average number of hours the boys spend on homework per month:

```
show 17 + 8 + 21 + 16 . . . . . . . .
```

```
show total number of hours for boys / number of boys
17
```

The boys spend an average of 17 hours a month on homework.

- Calculate the average number of hours the girls spend on homework per month:

```
show 16 + 14 + 13 + 16 + 18 . . . . .
```

```
show total number of hours for girls / number of girls
15
```

The girls spend an average of 15 hours a month doing homework.

Planning the Computer Version of Survey

It is essential that students think of the subjects for their surveys and the questions to be asked *before starting the project on the computer.*

Encourage the students to think of original subjects for their surveys, rather than simply copying the example given in the project booklet. If students have difficulty thinking of original ideas for their surveys, you may want to suggest ideas that relate to a current political/social situation or issue that you have discussed in class. Here are some examples:

Which candidate will win the next municipal election.

Which team will win a sports event.

The‸number of students who like taking the bus to school.

The number of students who think there should be more parks in their town or city.

The average height of the boys and the girls in the class.

The number of students who think the municipal government is handling a particular situation well.

Working Through the Project on the Computer

Questions and Answers

The **question** procedure asks the questions for the survey. It is similar to the **talk** procedure in *Conversations*, except that two questions are asked. The answers are named **hours** and **identity**.

Keeping a Permanent Record

The **survey** procedure:

- Runs **question**.
- Gets another page and prints the values of **hours** and **identity** on this page.
- Goes back to the original page (in the example given in the project, the original page is named survey).

- Confusion about why it is necessary to print **hours** and **identity** on another page.

  If LogoWriter doesn't print the values of **:hours** and **:identity** after each person has completed the survey, the values will change without the earlier values having been recorded.

  Go through an example with the students, showing them how LogoWriter assigns values to the variables **hours** and **identity**.

  When the first "respondent" completes the survey, LogoWriter asks:

```
How many hours of television do you watch each month?
```

  The respondent might answer **18**. LogoWriter gives this answer a name, **hours. Hours** is a variable whose value is 18.

  Then LogoWriter asks:

```
Are you a boy or a girl?
```

  The respondent might answer **boy**. LogoWriter names the answer **identity**. The value of **identity** is boy.

  Check the values of **hours** and **identity**:

```
show :hours
18
show :identity
boy
```

  The next respondent is asked the same questions. LogoWriter asks:

```
How many hours of television do you watch each month?
```

  The respondent might answer **10**. This answer is named **hours**. LogoWriter asks the second question:

```
Are you a boy or a girl?
```

  The respondent might answer **girl**.

To see the values of **hours** and **identity**:

```
show :hours
10
show :identity
girl
```

The values of **hours** and **identity** change every time someone answers the questions.

By printing the values of **hours** and **identity** on another page, the answers of each respondent are recorded.

- Running the **survey** procedure without naming the page on which the answers will be saved.

  The page that the answers will be recorded on must be named *before* the **survey** procedure is run; otherwise, LogoWriter will print the following message:

  ```
  There's no page named answers
  ```

  In this case, the student must get a new page, name the page answers, and go back to the page with the survey to run the **survey** procedure.

Starting Automatically

Students are shown how to define a **startup** procedure. The **startup** procedure gives the instructions to the person who is going to complete the survey. The **startup** procedure "starts" the survey the first time you get the page with the survey, and each time a respondent has completed the survey.

The Results

Encourage the students to use LogoWriter's arithmetic reporters to calculate the averages for their data. Direct them to the *LogoWriter Arithmetic* Reference Card (gray section of activity cards) for information on how to use the arithmetic reporters.

Bar Graphs

PAGE 15

This project involves drawing a bar graph to represent some information. Students who have completed *Taking a Survey* may be able to use the data from their survey for the graph. Alternatively, students can graph data for a history, science, geography or social studies project.

Note: The procedures provided in the project are designed for data that is plotted in intervals of 10. **Make sure students use data that can be plotted appro riately in intervals of 10 for their first graph.** You may want to provide data that is suitable.

Bar Graphs provides experience in:

Superprocedures and subprocedures.
Procedures with inputs.
Passing an input from one procedure to another.

Off-computer Activities

What is a Graph?

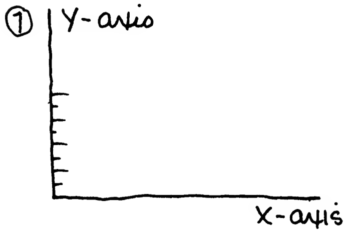
Graphs are widely used as a means of representing data. Make sure that the students understand what a graph is: a diagram representing a system of interrelations among two or more things. See if they can think of examples of kinds of information (data) that can be represented in graphs. Examples: average rainfall or temperature in different seasons, average heights of boys and girls at different ages, the population of a city at ten year intervals. Presenting information in a graph makes it easy to see the relation between different data.

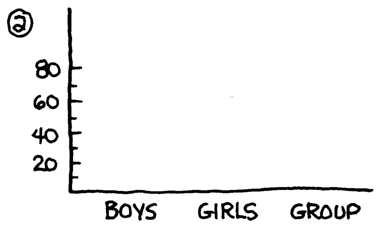
If you want to incorporate this project into a broader class discussion or project on graphing, you could also discuss different types of graphs (bar graphs, dot graphs, line graphs) and how to make them. The *Cartesian Grid* activity card (blue section) shows how to make a dot graph.

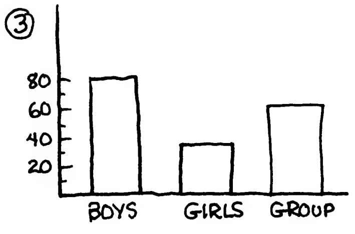
A Blackboard Version of a Graph

Draw a "blackboard version" of a graph before the students start the project.

1. **Draw the axes of the graph.** It is important that the students understand that the markers on the y-axis must be equally spaced and represent equal intervals. Use intervals of 10, as is shown in the project.
2. **Label the axes.**
3. **Draw the bars to represent the data.**







Working Through the Project on the Computer

Students should start by drawing a graph similar to one that is shown in the project. This means using data that is appropriately measured in intervals of 10.

Drawing a Graph

A **repeat** instruction is used to draw the y-axis and the markers on it.

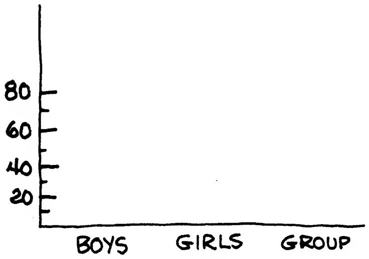
If the x- or y-axis is longer than the example given in the project, students may need to position the turtle towards the lower left side of the page to draw the axes. The **setpos** command can be used for this. For example:

```
setpos [-60 -50]
```

In this case the student will have to change some of the procedures presented in the project to reflect the new starting position.

Labeling the Graph

Students can label the axes either under procedure control or by using the **label** keys. The way the axes are labeled must reflect the nature of the data to be represented on the graph.



The Bars

The **brush** procedure uses a simple technique of stamping a short wide shape, moving the turtle **forward 1**, stamping the shape again, etc. The input to **brush** tells LogoWriter how many times to repeat these instructions.

Bar takes an input for **height**. The input to **bar** determines the height of the bar in terms of turtle steps.

The **enter** procedure positions the turtle along the x-axis and then draws a bar the height specified by the input **height**.

- Bars extending below the x-axis.

  The shape used in the **brush** procedure must be composed of *the middle row only* of the shape. If the shape extends below the x-axis, it will start stamping there.

- Confusion about how the input to **bar** is used by the **brush** procedure.

  Confusion is likely to stem from the fact that **brush** has an input named **times**, but when **bar** runs **brush**, the input to **brush** is **height**. The procedures in the project are as follows:

```
to brush :times
repeat :times [pd stamp pu fd 1]
end
```

```
to bar :height
brush :height
bk :height
end
```

  Explain what happens when LogoWriter runs **bar**. For example:

```
bar 20
```

  In this example, **:height** receives 20 as its value. Each time **:height** is encountered, it will be replaced with 20. **Brush** uses 20 as its input. It will run:

```
repeat 20 [pd stamp pu fd 1]
```

- Confusion about the height of the bars in relation to the markers on the y-axis.

  In the example used in the project, the markers on the y-axis are 10 turtle steps apart, and are labeled as representing intervals of 10 (hours). If the input to **enter** is 10, the bar is drawn to the 10 marker; if the input to **enter** is 40, the bar extends to the 40 marker, etc.

  If a student uses different intervals on the y-axis, the input to **enter** will have to be calculated according to the number of turtle steps. For example, if the bar should go up to the fifth marker on the y-axis, and the markers on the y-axis are 15 turtle steps apart, the student will have to multiply 5 x 15 and run **enter 75**.

  Remind students that they can use the LogoWriter arithmetic reporters to perform these calculations.

Topics for Discussion

Introduction 94

Turtle Geometry: The Turtle's State 95

Grammar: Instructions and Inputs 97

Defining Procedures 99

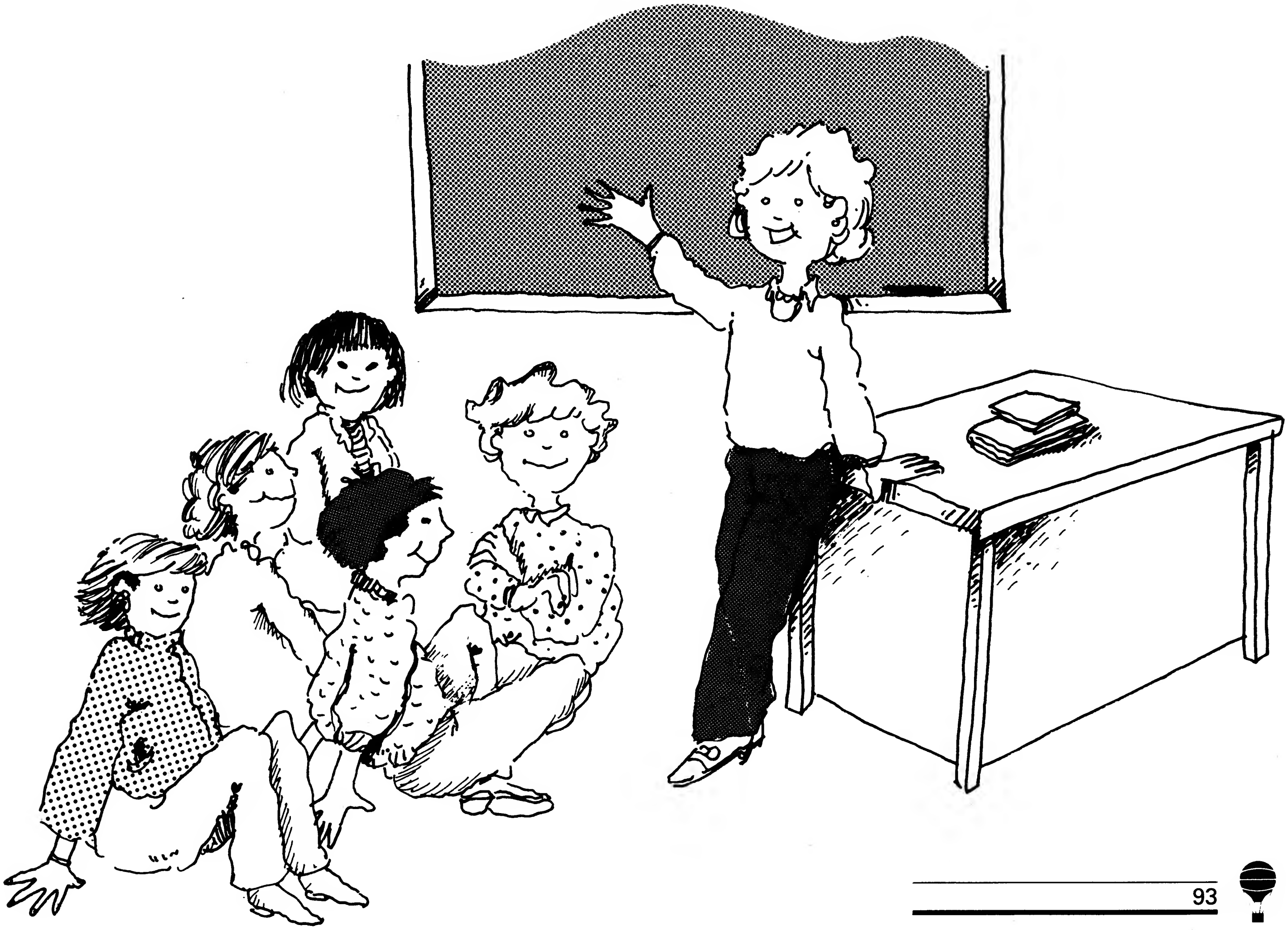
Superprocedures and Subprocedures 101

Commands and Reporters 103

Procedures With Inputs 105

Recursion 107

Variables 109



Introduction

This chapter contains outlines for class discussions on LogoWriter concepts. Each topic should be presented to the class or group of students when they are ready to use it in a project. These topics are presented in one chapter for easy reference. You will probably find it necessary to review each concept several times as the students use it in programming.

Two symbols are used throughout this chapter:

💡 Indicates information that is suitable for class discussion. It is information for students to "think about."

🎬 Indicates information that is suitable for class activities – both on and off-computer. Some of the information can be presented on the blackboard. At other times you may want to present the information while the students are sitting at the computers, so that they can try out the ideas on the computer.

The following chart indicates when each topic in this chapter should be introduced in relation to the projects.

	BEFORE	AFTER
On Your Mark, Get Set, Go		Turtle Geometry: The Turtle's State Grammar: Instructions and Inputs
Word Adventures		
Crossword Puzzle	Defining Procedures	Superprocedures and Subprocedures
Secret Code	Commands and Reporters	
Words Come Alive!		
Row, Row, Row Your Boat	Procedures With Inputs	
You and Your Friends		
Walking Through Your Neighborhood	Recursion	
Get the Facts!		
Conversations	Variables	

Turtle Geometry: The Turtle's State

The effect of **right** and **left** may be puzzling to students who have never done turtle graphics. While it is common sense that **forward 50** changes the turtle's position by 50 steps, a lot of people expect **right 50** to draw horizontal lines towards the right of 50 steps. Instead, **right 50** makes the turtle pivot 50 degrees to its right without changing its position.

The ingenuity of turtle geometry is that position and direction (heading) are two independent components of the turtle's state. **Forward** and **back** change the turtle's position; **right** and **left** change the turtle's direction. Since position and direction are independent, these components are easily controlled.

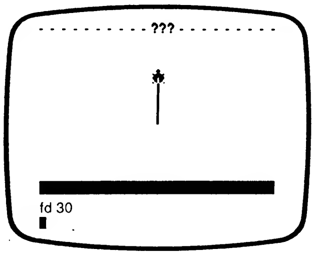
Point out to the students that when they are riding a bicycle they have to pedal to go forward, and turn the handle bars to turn. Once they become used to riding a bicycle, this becomes automatic. Learning how to control the turtle is a little like learning how to ride a bicycle. The student has to think about what he or she wants the turtle to do, and give it very specific instructions how to do it.

Students can model the turtle's state by "playing turtle." Ask one student to be the turtle. This student can stand at the front of the classroom, facing the others. Then ask the other students how they would tell the turtle to get to the classroom door using turtle instructions. They might give instructions such as the following:

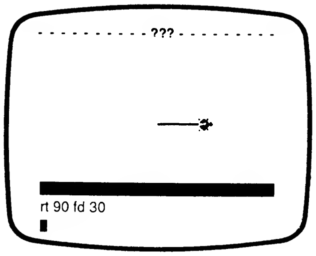
"forward 15 steps"
"right ⎵" (no need to be specific about the amount)
"forward 5 steps"
"right a bit more"
"forward 2 steps"

A more complicated example might be explaining how to go from the classroom door to the library.

Give examples (on the board or computer) of how the turtle's starting heading affects where a line is drawn on the page.

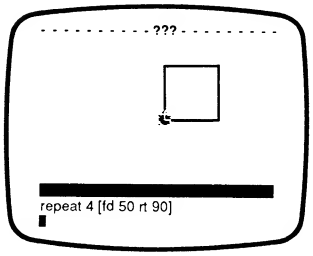


```
fd 30
```



```
rt 90 fd 30
```

Try:



```
repeat 4 [fd 50 rt 90]
```

Ask the students to predict where the square will be drawn with these instructions:

rt 90 repeat 4 [fd 50 rt 90]
lt 90 repeat 4 [fd 50 rt 90]
rt 45 repeat 4 [fd 50 rt 90]

Grammar: Instructions and Inputs

We use language as a means of communicating with other people. Computers don't understand English, so we have to use a language that people and computers understand. LogoWriter is a *computer language.*

LogoWriter is always waiting for instructions. It will do whatever you tell it to, as long as it understands the instructions. Show students examples of the way LogoWriter follows instructions.

On the blackboard, write:

cg

Ask the students what this command tells LogoWriter. **Cg** tells LogoWriter to clear the graphics from the page.

Try another example. What does this instruction tell LogoWriter?

setc 2

It tells LogoWriter to set the turtle to color 2.

What about this?

50

Ask a student to check on the computer. LogoWriter replies:

```
I don't know what to do with 50
```

LogoWriter has to be told what to do with 50, as in:

forward 50

Forward 50 is a LogoWriter instruction. It is composed of a *command* and an *input.*

forward 50

This is the command. This is the input.

Some commands need inputs, others do not. Ask the students to think of some commands that need inputs, and some that do not need inputs.

Pu is an example of a command that does not need an input. It tells the turtle to lift its pen up. LogoWriter does not need any other information. **Pu** is a complete instruction. In fact, **pu** won't accept an input. If you try giving **pu** an input, as in:

```
pu 50
```

LogoWriter replies:

```
I don't know what to do with 50
```

Look at the kinds of inputs commands take:

Forward only takes numbers as its input. It won't accept any other kind of input. If you try giving it another kind of input, as in:

```
forward [a lot]
```

LogoWriter replies:

```
forward doesn't like [a lot] as input
```

Show the students other examples:

Repeat takes a number as its first input and a list as its second input.

repeat 6 [fd 50 rt 60]

Repeat this many times. This list of instructions.

A list is always surrounded by square brackets. Square brackets are like an envelope. The instructions to be repeated go inside the envelope. In the above example, the instructions **fd 50 rt 60** will be repeated 6 times.

Print and **show** take a number, a word, or a list as input.

```
show 5
```

```
print [Welcome to LogoWriter!]
```

```
print "hi
```

The quotation mark is used to indicate that **hi** is an input, rather than a LogoWriter instruction. Look what happens if you forget it:

```
print hi
```

LogoWriter replies:

```
I don't know how to hi
```

LogoWriter thinks that hi is a command and tries to run it. It realizes that hi is not in its vocabulary.

Defining Procedures

To get the computer to do something in LogoWriter, we have to use words that LogoWriter understands. LogoWriter has a "built-in" vocabulary of words that it knows. These are called *primitives*. **Forward**, **cg**, **pu**, **setc** are examples of primitives.

You can add words to LogoWriter's vocabulary. Words that you add to LogoWriter's vocabulary are called *procedures*. A procedure is just a group of LogoWriter instructions with a name. Procedures work just like primitives.

You can draw the following analogy:

A dictionary contains a list of words in a language. These words have a meaning; they are defined for us. Imagine if we could add to the dictionary, making up new words and defining their meanings. We could simply turn to a special place in the dictionary, write in the name of the word and its definition. Once we had done this, we could use this word like any other word in the dictionary.

We can think of LogoWriter's vocabulary in this way. Some words are already there, defined for us. These are called primitives. But we can add other words to LogoWriter's vocabulary that are defined in terms of words that already exist in the language. These words that we define ourselves are called procedures. They can be used just like primitives.

Introduce the technique of defining procedures. Tell the students to follow these steps:

- Press the **flip** keys to go to the flip side of the page.

Point out the physical differences between the front side and the flip side of the page. The front side has a narrow dotted line across the top. The flip side has a solid wide band across the top. ***Flip Side*** is printed in the top right corner.

- Type in the procedure definition. Example:

```
to rabbit                              Name of the procedure.
pu                          ──┐
fd 40                         │
label [My rabbit Fuzzy]       ├── Body of the procedure
bk 40                         │   definition.
setsh 22                    ──┘
end                                    End of the procedure.
```

To run the procedure, press the **flip** keys to flip to the front of the page, and type the name of the procedure in the command center.

```
rabbit
```

Logo Writer runs each line of the procedure. Typing **rabbit** in the command center is the same as typing each line of the procedure in the command center.

LogoWriter has two *special words* for defining procedures, **to** and **end**. A procedure definition always begins with **to** followed by the name of the procedure. The name of the procedure is always one word. It can be anything except a number or a primitive. In the example given above, the procedure is named **rabbit**. **End** is used to indicate the end of the procedure definition. The instructions between **to** *procedure name* and **end** are the body of the procedure definition.

Compare the LogoWriter page with a real page. The front of the page is like a display area. It is for the "end product." The flip side of the page is for defining procedures.

When a procedure is defined, it becomes part of the LogoWriter vocabulary for that page. On any particular page, the LogoWriter vocabulary available consists of LogoWriter's primitives plus the defined procedures *on the flip side of that page*. The flip side of each page becomes a "mini-dictionary" for that page. Each time the student turns to a new page, a whole new vocabulary can be defined for that particular page.

Superprocedures and Subprocedures

There are two ways to run a procedure:

- By typing its name in the command center.
- By running the procedure inside the definition of another procedure.

Show the students a simple example of how a procedure can be run by another procedure. Define a **square** procedure:

```
to square
repeat 4 [forward 40 rt 90]
end
```

Then define a **manysquares** procedure to make a design of squares:

```
to manysquares
repeat 9 [square rt 40]
end
```

When you run **manysquares**, you are running the procedure **square** indirectly. **Manysquares** runs it for you.

Manysquares is the *superprocedure* that runs the *subprocedure* **square**.

Give students a **triangle** procedure. Challenge them to find ways to use it as a subprocedure.

```
to triangle
repeat 3 [fd 40 rt 120]
end
```

Examples:

```
to windmill
repeat 4 [triangle rt 90]
end
```

Windmill is the superprocedure, **triangle** is the subprocedure.

```
to house
square
fd 40
rt 60 triangle
end
```

House is the superprocedure, **square** and **triangle** are the subprocedures.

Commands and Reporters

There are two types of LogoWriter primitives: *commands* and *reporters*.

Commands tell LogoWriter to do something. Show examples of commands:

forward	***Tells*** the turtle to move forward some amount.
pu	***Tells*** the turtle to lift its pen.
getpage	***Tells*** LogoWriter to get a page.
print	***Tells*** LogoWriter to print something on the page.

There are two ways for a command to get an input:

- Directly, as in

  ```
  print "hi
  ```
  or
  ```
  print [I love LogoWriter!]
  ```

- Indirectly, as in

  ```
  print color
  ```

Color is a *reporter.** Reporters "report" information – they deliver information about something. In this case, **color** reports the color of the turtle to **print**.

Reporters always report their information to a command. Show students what happens if a reporter is used by itself.

```
color
```

LogoWriter replies:

```
I don't know what to do with 1
```

Color got the information – the color of the turtle, but didn't know what to do with it.

You can draw the following analogy. A reporter's job is to get information about something. When the reporter has done his or her job (found the information), he or she delivers this information to someone. The person the reporter gives the information to uses it for something, for instance, to print in a newspaper article or broadcast on television.

Just as LogoWriter has commands to do all kinds of things, it also has reporters to do all kinds of things. It has reporters to report on the number of characters on the page, arithmetic functions, the turtle's position on the screen, the way the turtle is heading, the shape the turtle is set to. Examples:

```
print [Reporters give lots of information]
show textlen
```

Reports the number of characters on the page.

```
show pagelist
```

Reports the pages you have created that are listed in Contents.

```
show 5 + 7
```

Reports the sum of its inputs.

```
show 10 * 12
```

Reports the product of its inputs.

* In other versions of Logo, the term "operation" is used instead of reporter.

```
show shape
        ↑
```

Reports the number of the shape the turtle is set to.

See the *Quick Change* activity card (red section) for examples of different reporters.

Procedures With Inputs

To introduce procedures with inputs, start with a concrete example. A good example to begin with is squares of different sizes.

Ask students how to define a procedure to draw a square with sides of 30 turtle steps. They will probably suggest a method such as the following:

```
to square
fd 30 rt 90
fd 30 rt 90
fd 30 rt 90
fd 30 rt 90
end
```

Now ask them how they would draw a larger square – say a square with sides of 60 turtle steps. They could define a **largesquare** procedure like this:

```
to largesquare
fd 60 rt 90
fd 60 rt 90
fd 60 rt 90
fd 60 rt 90
end
```

Now suppose they wanted to draw a tiny square (say for a door knob on a house they are drawing). They would have to define yet another procedure.

It quickly becomes evident that it would be convenient to have a **square** procedure that could draw a square of any size. This can be done by defining a procedure that takes an input.

Show the students how to define a square procedure that takes an input for size. Call this procedure **newsquare**. Go through the steps based on their old **square** procedure.

square procedure

```
to square
fd 30 rt 90
fd 30 rt 90
fd 30 rt 90
fd 30 rt 90
end
```

newsquare procedure

to newsquare
fd *how much* – the amount will depend on the size of the square. Invent a name for this amount, say **size**.

```
fd :size rt 90
fd :size rt 90
fd :size rt 90
fd :size rt 90
end
```

Put a colon in front of size to indicate "the thing" called size.

The name of the input must appear on the title line of the procedure, to inform LogoWriter that the procedure must have an input.

The **newsquare** procedure will look like this:

```
to newsquare :size
fd :size rt 90
fd :size rt 90
fd :size rt 90
fd :size rt 90
end
```

Show the students what happens if they type

```
newsquare
```

in the command center.

LogoWriter prints the message:

```
newsquare needs more inputs
```

Just as **forward** needs an input to know how far forward to go, **newsquare** needs an input to know the size. Try:

```
newsquare 50
```

LogoWriter gives **size** the value 50. Follow through the procedure one line at a time to show students how LogoWriter runs it.

```
to newsquare :size           newsquare 50
fd :size rt 90               fd 50 rt 90
fd :size rt 90       ⇒       fd 50 rt 90
fd :size rt 90               fd 50 rt 90
fd :size rt 90               fd 50 rt 90
end
```

Try different inputs:

newsquare 80	Draws a large square.
newsquare 10	Draws a tiny square.

When a procedure is defined with an input, the value of the input is unknown. It is only when the procedure is run that the value of the input is known. For this reason, the input must always be given a name. In the **newsquare** procedure, **size** is the name of the input.

The name of an input must always be preceded by a colon. ***The colon is used to indicate that the word following it is an input rather than a procedure***. The colon means "the thing" called **size**. The value of **size** is determined at the time the procedure is run.

In the **newsquare** procedure:

newsquare 50

50 is the thing called **size**.

Using meaningful names for the inputs to procedures makes it easier to remember what the input does. For example:

to rectangle :height :width

indicates that **rectangle** needs two inputs. The first input determines the height and the second the width of the rectangle. This is much more meaningful than:

to rectangle :x :y

Recursion

Recursion is one of the most powerful aspects of LogoWriter. It is important that recursion be well explained, using examples that are easy to understand.

Students should already have mastered the idea of superprocedures and subprocedures. Recursion should be introduced as a procedure that runs itself as a subprocedure.

For younger students, you may want to introduce recursion with a group activity that mirrors simple recursion. This activity is helpful in understanding simple recursive procedures such as **move** (in *Walking Through Your Neighborhood*). It consists of having the students do something whenever they hear a certain word. For example:

Each time you hear the word hop, hop forward.

You could write this as a procedure:

```
to hop
hop forward
end
```

Now play hop.

teacher	students
"hop"	hop forward

Show the students how to make a modification to hop. After hopping forward, they should tell themselves to hop. What happens?

teacher	students
"hop"	hop forward
	"hop"
	hop forward
	"hop"
	hop forward
	"hop"
	and so on...

Show students how to define a recursive procedure. The hop example above can be adapted to give a similar effect on the computer.

```
to hop
fd 10
label [I love to hop!]
wait 5
hop
end
```

Let students try it on the computer:

```
pu
hop
```

To stop the procedure, press the **stop** keys.

Here is another example:

```
to drive
fd 1
drive
end
```

Try:

```
pu
rt 90
setsh 27
drive
```

To stop the procedure, press the **stop** keys.

LogoWriter runs one line of a procedure, then goes on to the next line of the procedure and runs it, and so on, until the end of the procedure. Demonstrate to the class how LogoWriter would run the **hop** procedure. The first line of the procedure gives the instruction to move the turtle forward 10 steps. The second line of the procedure gives the instruction to label ***I love to hop!.*** The third line of the procedure gives the instruction to **wait 5**. The next line of the procedure gives the instruction to run **hop**. The first line of **hop** tells the turtle to move forward 10 steps, the second line of the procedure gives the instruction to label ***I love to hop!,*** etc.

Discuss the meaning of recursion. Tell this recursive riddle:
If you had two wishes, what would your second wish be?
Answer: Two more wishes.

Recursion is something that goes on infinitely. See if the students can think of things in real life that are recursive. Examples:

- Days of the week.
- Seasons of the year.
- Months of the year.
- Waves in the ocean.

Variables

Naming things is an intrinsic part of LogoWriter. You might want to start by asking students what can be named. They will probably mention pages and procedures. There is another aspect to naming. In LogoWriter you can name a thing, or a group of things.

Show the students a simple example of how to name something:

$$\textbf{name } \underbrace{\textbf{[horse cat dog]}}_{\text{The things to be named.}} \ \underbrace{\textbf{"animals}}_{\text{The name.}}$$

LogoWriter remembers what the name **animals** stands for. Try

```
show :animals
```

Animals is a variable whose value is **horse cat dog**.

The colon means "the thing" called animals.

LogoWriter prints what **animals** stands for:

```
horse cat dog
```

Give students a few other examples to try out:

```
name [Jennifer Nicole Peter Zachary] "friends
print :friends
Jennifer Nicole Peter Zachary

name [spring summer winter autumn] "seasons
print :seasons
spring summer winter autumn
```

Creating variables is useful when you want LogoWriter to remember something. Here are some useful ways to use variables:

- To name the turtles. Examples:

```
name [0 1] "team1
name [2 3] "team2

tell :team1
st fd 50

tell :team2
st bk 30
```

- To name colors. Examples:

```
name ⌴ "red
name ⌴ "blue
```

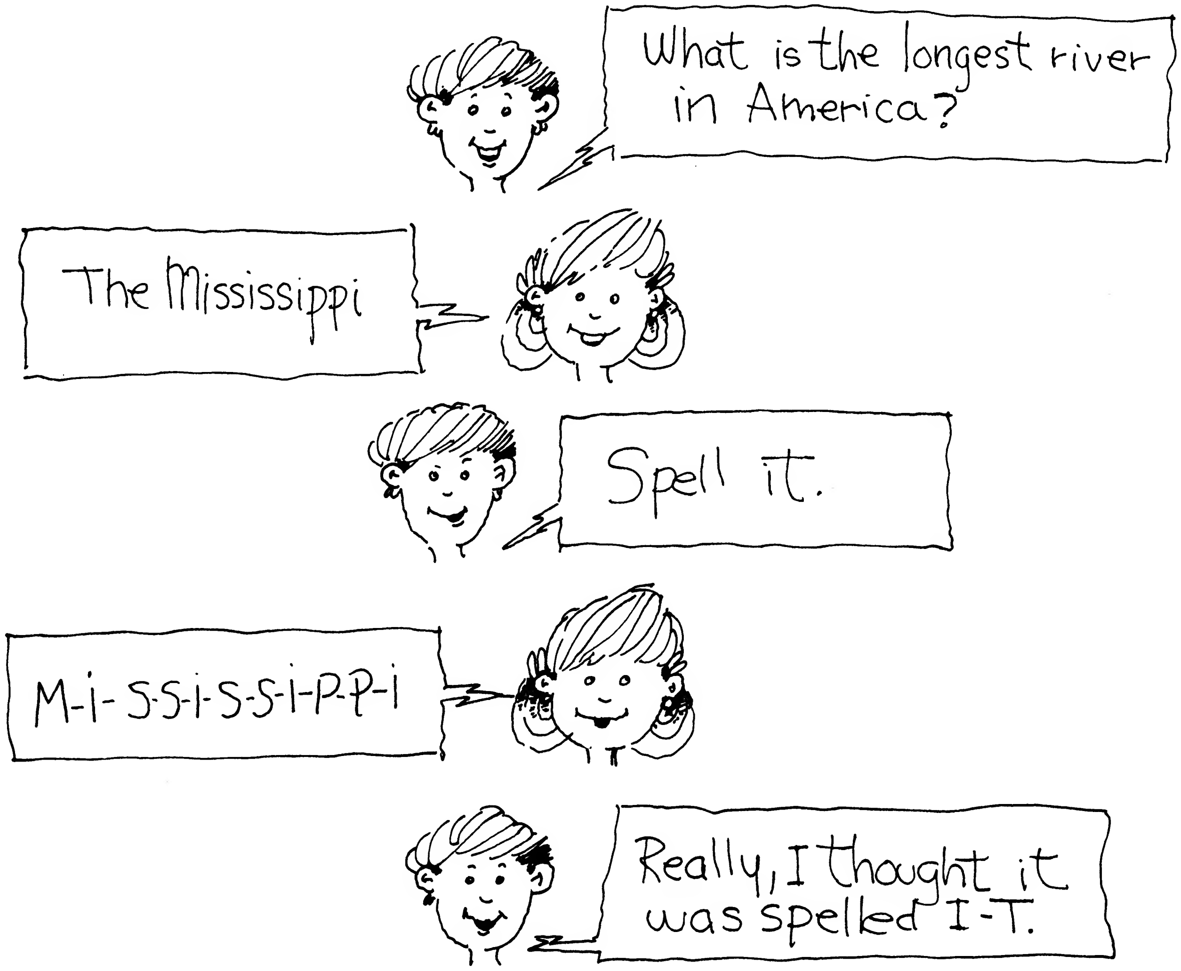
The color number.

```
tell 0 setc :red
tell 1 st setc :blue
```

- To name shapes. Examples:

```
name 14 "heart
name 26 "truck

setsh :heart
setsh :truck
```

The function of the quotation mark and the colon is often confusing. The following riddle is familiar to many students. It helps illustrate the difference between the colon and the quotation mark.


What is the longest river in America?
The Mississippi
Spell it.
M-i-s-s-i-s-s-i-p-p-i
Really, I thought it was spelled I-T.

People usually understand which of these you mean from the tone of your voice. If you write it down, punctuation marks are used to show the difference. For example:

Spell it means spell the thing you're talking about.

Spell "it" means spell "it" itself.

LogoWriter needs punctuation marks too. Colons and quotation marks are used.

The quotation mark says that you mean the word itself.

The colon says that you mean what the word stands for.

Index

A

activity cards 11, 12, 20
  chart of 14-16
all 65, 68
animation 50
arithmetic reporters 44, 87

B

bg 60

C

cb 79
cf 43
charunder 79
colon 106, 109, 110
color 60
colorunder 70, 71
command center 32
commands 103
communication arts 12, 77
computer labs 24
conditional instructions 70, 73, 79
Contents page 28, 32
copying shapes 52, 67
creating shapes 51, 55
ct 43
cursorpos 79, 80

D

defining procedures 37, 41, 43, 51, 55
  technique of 99
delete 43
disk
  LogoWriter 11, 13

E

each 65, 68
end 39, 99

F

fill 28, 31, 40
flip keys 39, 99

G

getpage 41
getting pages under procedure control 42

H

Help page 34
history 61, 79
home 61

I

if 60, 70
ifelse 84
illustration of text 50
inputs 97
insert 43, 45
instructions 97
integrating text and graphics 77, 79
interactive procedure 85

K

keyboard
  posters 11, 29
  stickers 11, 29

L

lab session 24
label 62
label text 29, 30
language arts 12, 36, 50
Learn pages 29
  erasing from the scrapbook 34
LogoWriter page 32
LogoWriter program 11
LogoWriter Reference Guide 11

M

mathematics 12, 60
messages 34
multiple turtles 51, 55

N

name 85, 109
names (variables) 85, 109
newpage 85

P

- **paddle** 84
- pages
  - changing under procedure control 41, 42
  - erasing 34
  - flipping 99
  - naming 33
  - saving 33
- posters, keyboard 11, 25
- primitives 29
- **print** 51
- printing on a printer 40
- **printscreen** 37
- probability 65
- procedures with inputs 55, 61, 65, 77, 90
  - passing an input from one procedure to another 90, 92
- project booklets 11
  - themes of 12
- projects, chart of 14–16

Q

- quotation mark 110

R

- **random** 65, 67
- **readchar** 84
- **readlist** 85
- recursion 107
- Reference cards 13, 25
- regular text 30
- reporters 103

S

- scrapbook concept 32
- screen 55
- **search** 79
- **sentence** 85
- **setpos** 70, 72
- **shade** 28, 31
- **shape** 60
- shape numbers 52
- Shapes page 32, 51
- **show** 43
- social studies 60
- startup as a page name 76
- **startup** procedure 60, 77, 79
- **stop** 70
- stop keys 107
- superprocedures and subprocedures 51, 55, 61, 65, 70, 90, 101

T

- **tab** 85
- **tell** 51
- text editing 41
- text editing primitives 43
- **textlen** 43
- **to** 38, 99
- **tone** 51
- **top** 43
- topics for discussion, chart of 94
- turtle graphics commands 30
- turtle heading 31
- turtle numbers 52
- tutorials 25

V

- variables 109

W

- **wait** 55
- **who** 60
- word processing 30, 43

LogoWriter is an Educational Product from Logo Computer Systems Inc.

LCSI®

PRINTED IN USA 8-93